ROM-Listing CPC 464/664/6128

Jörn W. Janneck
Till Mossakowski

ROM-Listing
CPC 464/664/6128

Ausführlich dokumentiertes Listing
aller Betriebssystem- und
BASIC-Routinen.
Detaillierte Hintergrundinformationen
zu Speicheraufteilung, Z80,
Video-RAM, Schaltplänen.

Die Unterschiede zum CPC 664/6128-
BASIC-Betriebssystem werden
gesondert kommentiert.

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

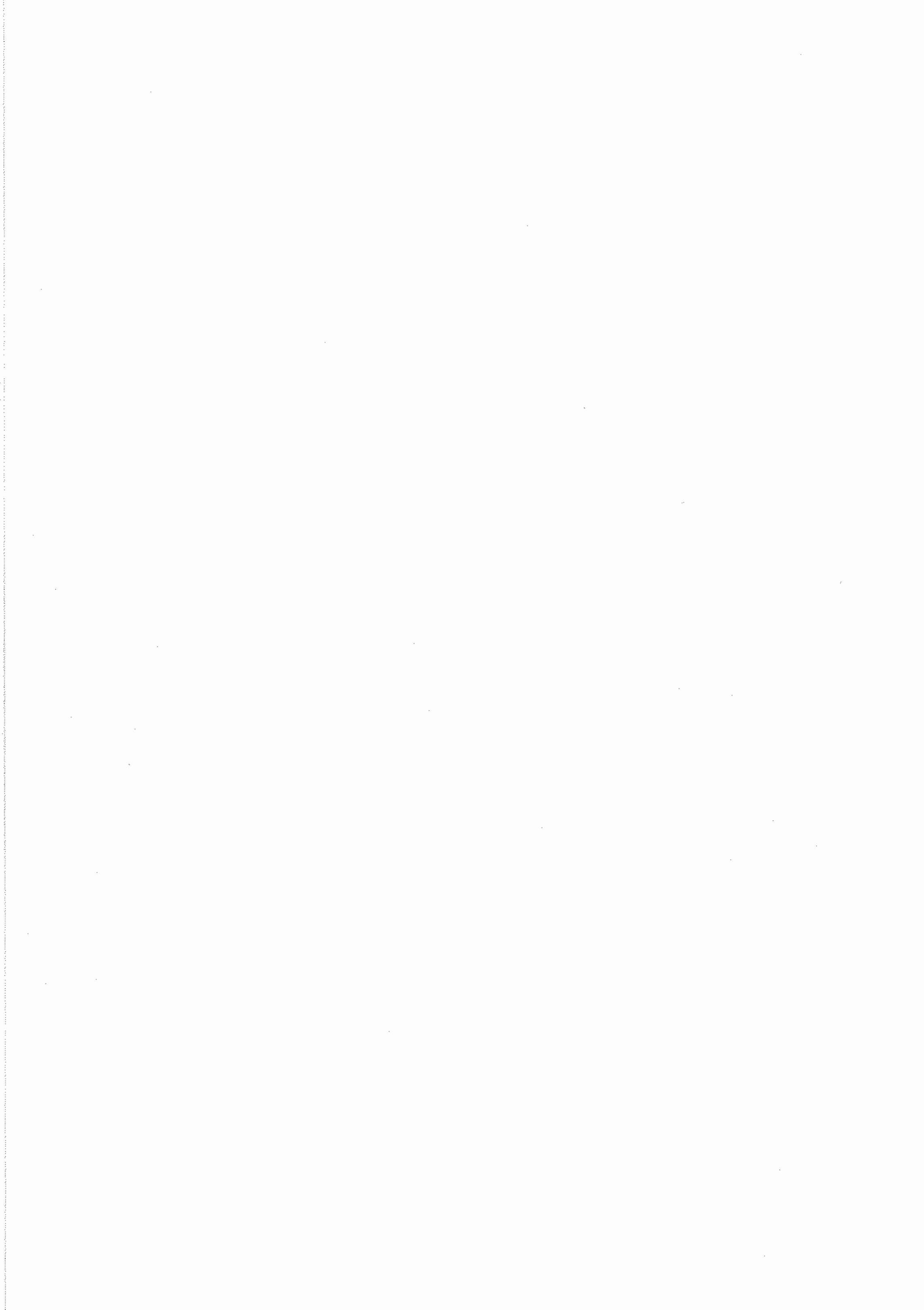
Janneck, Jörn W.:
ROM-Listing CPC 464, 664, 6128 : ausführl. dokumentiertes Listing aller
Betriebssystem- u. BASIC-Routinen ; detaillierte Hintergrundinformationen zur
Speicheraufteilung, Z 80, Video-RAM, Schaltpl. ; d. Unterschiede zum
CPC 664/6128-BASIC-Betriebssystem werden gesondert kommentiert / Jörn W. Janneck ; Till Mossakowski. —
Haar bei München : Markt-und-Technik-Verlag, 1985
ISBN 3-89090-134-4
NE: Mossakowski, Till:

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
89 88 87 86

ISBN 3-89090-134-4

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

Inhaltsverzeichnis



Vorwort

Dieses Buch bietet Ihnen eine Fülle von Informationen, die einfach unerläßlich sind, wenn Sie Ihren Computer nicht nur als Black Box benutzen wollen, sondern seinen Aufbau kennenlernen möchten. Nur so sind Sie in der Lage, die Fähigkeiten Ihres Systems voll auszuschöpfen. Dies gilt besonders dann, wenn Sie gerade anfangen, in Maschinensprache zu programmieren.

Das ROM des Schneider CPC stellt eine gut nutzbare Bibliothek von Unterprogrammen dar, deren effiziente Nutzung jedoch genaue Informationen erfordert. Diese Informationen haben wir in dem vorliegenden Buch zusammengestellt. Darüber hinaus erschien es uns zu einem umfassenden Verständnis notwendig, die Details und Hintergründe näher zu erläutern.

So beginnt das Buch mit einer genauen Beschreibung der Hardware des Schneider Computers. In diesem Teil haben wir alle Chips, die für den Programmierer von Interesse sind, eingehend besprochen und ihre Funktionen und Programmierung dargestellt.

Im zweiten Teil werden Sie dann in einige - für den CPC relevante - Strukturen und Techniken eingeführt. Die Darstellung baut auf Strukturen auf, die aus Basic bereits bekannt sind, und ist auch über die Grenzen des CPC hinaus ganz allgemein für das Entwerfen und Verstehen von Software wichtig.

Die beiden folgenden Teile sind ganz der genauen Erläuterung der Firmware des CPC gewidmet. Die wichtigsten Grundlagen und Zusammenhänge des Betriebssystems und des Basic sind hier zusammengefaßt und erklärt. Dies beginnt bei der Beschreibung der wichtigen Routinen und geht bis zur Darstellung der mathematischen Grundlagen der Fließkomma-Operationen.

Es schließt sich der fünfte Teil mit Tabellen zu den ROM-Listings an, auf den besonders der Maschinen-Programmierer ständig zurückgreifen wird. Sie fassen alle wichtigen Einsprünge in das Basic und das Betriebssystem auf einen Blick zusammen und werden so zu unentbehrlichen Werkzeugen.

In Teil sechs folgen dann die Listings des Basic- und Betriebssystem-ROMs des CPC 464 sowie eine Zusammenstellung der Änderungen beim CPC 664 und CPC 6128. Zum vollständigen Verständnis und zur Nutzung der ROMs stellen diese Listings unschätzbare Dokumente dar. Sie sind vollständig dokumentiert und jede Routine ist mit ihrer Parametrisierung dargestellt.

Den Abschluß bilden dann die Anhänge, mit allen wichtigen Daten über die Bausteine des Schneider Computers, sowie der Schaltplan des CPC.

Wir hoffen, mit unserem Konzept ihren Wünschen gerecht geworden zu sein. Für Anregungen und Kritik sind wir jederzeit offen und dankbar.

Bremen, Juli 1985

Die Autoren

Danksagungen

Wir danken allen, die - direkt oder indirekt - zur Herstellung dieses Buches etwas beigetragen haben. Besonderen Dank schulden wir jedoch Michael Reimann, der uns seinen Zorn nicht spüren ließ, als wir nach vielen Übertragungen von Daten schließlich den Drucker-Port seines Computers vernichteten. Desweiteren danken wir Christian Holsten, Stephan Bechtold, Martin Marschner, Stefan Runge, Jörn Volkers, Axel Damm, Mathias Bremer und Nicolas Richter, ohne deren Eigentum dieses Buch jetzt nicht in Ihren Händen läge. Unser Dank gilt ebenso Helmut Tischer für die Durchsicht des Manuskripts sowie den Schneider Rundfunk Werken, Türkheim, für die Unterstützung mit wichtigen Informationen. Auch unserem Umfeld, insbesondere unseren Familien, schulden wir Dank, da es mit Geduld und ohne Klagen die Lasten ertrug, die wir ihm durch unsere Arbeit auferlegten.

1 Hardwarebeschreibung

1.1 Prozessor Z80A

1.1.1 Einführung

Der Z80 von Zilog ist ein weit verbreiteter 8-Bit-Mikroprozessor, der auch das Herzstück des CPC bildet (siehe Anhang A1.2). Wir wollen hier nicht auf die Programmierung dieses Prozessors eingehen, da dies allein schon ein Buch füllen würde. Wir wollen vielmehr kurz einen Überblick über die Möglichkeiten dieses Chips geben, wobei wir grundlegende Kenntnisse in der Maschinenprogrammierung voraussetzen.

Der Prozessor ist das Gehirn eines Computers, ist der Baustein, der die Programme ausführt. Innerhalb des Computers arbeitet er zu diesem Zweck mit Komponenten zur Speicherung und Ein-/Ausgabe von Programmen und Daten zusammen. Die Kommunikation mit diesen Einheiten wird über drei voneinander getrennte Sätze von Leitungen abgewickelt, den "Bussen". An einen Bus sind immer alle Einheiten angeschlossen, was ihn zu einer effektiven Möglichkeit deren Verbindung macht. Grundsätzlich unterscheidet man in einem Computer wie dem CPC drei Busse: Adreß-, Daten- und Steuerbus.

Der Speicherbereich dieses Computers ist eingeteilt in Speicherstellen, von denen jede genau ein Byte speichern kann. Diese Speicherstellen sind durchnummeriert und ihre jeweilige Nummer nennt man Adresse. Der Prozessor kann auf ein Byte zugreifen, indem er dessen Adresse über den Adreßbus an die Speicherbausteine sendet. Diese lesen daraufhin das adressierte Byte und geben es aus bzw. schreiben ein von ihm gesandtes Byte in die entsprechende Speicherstelle. Das Byte selber wird auf dem Datenbus übertragen. Hier läßt sich erkennen, daß der Datenbus in zwei Richtungen benutzt werden kann (vom und zum Prozessor, man nennt dies bidirektional), während der Adreßbus nur Signale vom Prozessor an den Speicher überträgt (unidirektional). Um nun aber festzulegen, in welche Richtung er den Datenbus benutzen möchte, gibt der Prozessor zusammen mit der Adresse noch weitere Signale auf zwei Leitungen des Steuerbusses (der Schreib- und der Leseleitung) aus. Neben diesen Signalen umfaßt der Steuerbus einige andere Signale, die zur Steuerung der Aktivitäten auf den Bussen dienen - z.B. den Systemtakt.

Ebenso wie der Speicherbereich ist auch der Ein-/Ausgabebereich in Bytes aufgeteilt und durchnumeriert. Der Zugriff kann also auch hier mit Hilfe von Adreß- und Datenbus erfolgen. Um nun zu vermeiden, daß sich Speicher- und Ein-/Ausgabebausteine gleichzeitig von einer Adresse auf dem Adreßbus angesprochen fühlen, gibt es zwei Signale des Steuerbusses, die zur Auswahl des entsprechenden Bereiches dienen.

1.1.2 Aufbau und Register des Z80A

Der Z80A besitzt interne Speicherstellen, Register genannt, die einen schnelleren Zugriff erlauben als der in zusätzlichen Chips vorhandene externe Speicher. Sämtliche Operationen im Z80A laufen über ein oder mehrere Register ab. Soll zum Beispiel der Inhalt einer Speicherstelle in eine andere übertragen werden, so kann dies nur in zwei Schritten geschehen: Der Inhalt der ersten Speicherstelle muß in einem Register zwischengespeichert und dann vom Register in die zweite Speicherstelle geschrieben werden.

Der Z80A besitzt 22 verschiedene Register mit zum Teil sehr unterschiedlichen Aufgaben.

- o Der Inhalt des Programmzählers (PC) ist die vom Prozessor während der Programmausführung gerade abgearbeitete Adresse. Da er eine Adresse enthält, ist er ein 2-Byte- bzw. 16-Bit-Register.

- o Der Akkumulator (A) ist ein 8-Bit-Universal-Register. Die meisten arithmetischen und logischen Operationen laufen unter Zuhilfenahme des Akkumulators ab. Da er für viele Operationen unerläßlich ist, ändern sich Inhalt und Bedeutung des Inhalts des Akkumulators sehr oft.

- o Ebenso wie der Akkumulator, sind auch die Register B, C, D, E, H und L 8-Bit-Register. Mit ihnen können jedoch nur ein Teil der mit dem Akkumulator möglichen Operationen ausgeführt werden. Sie behalten ihre Werte daher auch meist über eine längere Zeit als der Akkumulator.

- o Die Register B, C, D, E, H und L können auch paarweise als 16-Bit-Register BC, DE und HL benutzt werden. Neben 16-Bit-Datenwörtern dient besonders das Register HL auch zur Speicherung von Adressen. Ein 16-Bit-Wort kann über diese Register leicht in zwei einzelne Bytes zerlegt oder aus ihnen aufgebaut werden.

- Ein Register, das weder Daten noch Adressen aufnimmt, ist das Flag-Register (F). Ein Flag (deutsch: "Flagge") dient dazu, einen besonderen Zustand anzuzeigen. Jedes Flag im F-Register kann nur zwei Zustände annehmen und benötigt daher nur ein Bit. Von den acht Bits im F-Register werden lediglich sechs Bits genutzt. Wegen dieser bitweisen Einteilung wird das F-Register selten als ganzes angesprochen. Die einzelnen Flags innerhalb des F-Registers haben eigene Bezeichnungen.

  - Das Carry-Flag (C oder CY abgekürzt) zeigt einen Übertrag bei Additionen und Subtraktionen zum nächst höheren Byte an und wird bei bitweisen Verschiebe-Operationen als Zwischenspeicher für einzelne Bits gebraucht. Es ermöglicht generell die Bearbeitung von Zahlen, die aus mehreren Bytes bestehen (z.B. REAL-Zahlen), in (durch die Registergrößen begrenzten) Schritten von einem oder maximal zwei Byte.
  - Das Halfcarry-Flag (H) zeigt einen Übertrag von Bit 3 zu Bit 4 bei einer Addition oder Subtraktion an. Zusammen mit dem N-Flag, das angibt, ob die letzte arithmetische Operation eine Addition oder eine Subtraktion war, und dem Befehl DAA (*D*ecimal *A*djust *A*ccumulator) kann so einfach mit BCD-Zahlen gerechnet werden. Im Basic-ROM des CPC werden diese Flags jedoch nur an einer einzigen Stelle benutzt.
  - Das Parity-Flag (P) und das Overflow-Flag (V) teilen sich ein Bit im F-Register. Wann dieses Bit entsprechend der Parität, also bei ungerader Anzahl von "1"-Bits, gesetzt ist und wann es einen Überlauf bei einer Zweierkomplement-Operation bedeutet, kann nur dem Programmzusammenhang entnommen werden.
  - Das Zero-Flag (Z) ist immer dann gesetzt, wenn das Ergebnis der letzten Operation gleich null war. Bei Vergleichen ist es dann gesetzt, wenn die Differenz der zu vergleichenden Größen null ist, also Gleichheit besteht.
  - Das Sign-Flag (S) stellt schließlich das Vorzeichen (wenn im Zweierkomplement gerechnet wird) bzw. das höchstwertige Bit des Ergebnisses der letzten Operation dar.

- Die Register A, F, B, C, D, E, H und L sind im Z80A doppelt vorhanden. Zwischen diesen beiden Registersätzen kann mit Hilfe zweier Befehle umgeschaltet werden. Den zweiten Registersatz des Z80A sollte man im CPC jedoch nicht benutzen, da er schon für die Speicherverwaltung (Banking) und Interrupts benötigt wird.

o Zwei weitere 16-Bit-Register sind die Indexregister IX und IY. Wie der Name Indexregister schon sagt, enthalten sie meist Adressen. Zeigen diese Adressen z.B. auf den Anfang einer Tabelle im Speicher, so können deren Werte durch Addieren von festen Zahlen zum Indexregister vor Generieren der endgültigen Adresse (z.B. IX+04) ausgelesen werden.

o Das Refresh-Register (R), ein 8-Bit-Register, dient dazu, die dynamischen RAMs in bestimmten Zeitabständen fortlaufend zu "refreshen". Bei einem Refresh werden die Kondensatorladungen erneuert, die die Information speichern. Der Inhalt des R-Registers wird vom Prozessor fortlaufend weitergezählt. Das Register wird aufgrund dieser Eigenschaft beim Schreiben und Lesen des Kassettensignals als Zeit-Zähler benutzt.

1.1.3 Funktionsweise des Stacks

Der Stackpointer (SP) ist ebenfalls ein 16-Bit-Register. Mit diesem Register ist es möglich, Unterprogramme zu realisieren. Ein Stack (Stapel) ist so aufgebaut, daß die zuletzt eingegebenen Daten zuerst wieder ausgegeben werden, was sich gut mit einem Stapel veranschaulichen läßt. Die Eingabe von Daten auf den Stack nennt man kurz auch "auf den Stack legen" oder "pushen" (engl. to push = schieben). Analog können die Daten dann wieder vom Stack geholt bzw. "gepopt" werden. Realisiert wird diese Datenstruktur durch den Stackpointer SP. Er zeigt stets auf das oberste Element im Stack und wird beim Pushen und Popen herunter- bzw. heraufgezählt.

Durch einen Stack sind daher auch verschachtelte Unterprogramme möglich: Beim Unterprogrammaufruf legt der Prozessor die Rücksprungadresse auf den Stack, während sie beim Rücksprung wieder vom Stack heruntergenommen wird. Bei verschachtelten Unterprogrammaufrufen stapeln sich so die Rückkehradressen auf dem Stack. Es können aber nicht nur Rücksprungadressen auf den Stack gelegt werden, auch für Register ist der Stack ein bequemer Zwischenspeicher. Die Datenspeicherung auf dem Stack wird in Kapitel 2.2.1 genauer beschrieben.

1.1.4 Interrupts

Ein Interrupt (Unterbrechung) ist ebenfalls ein Unterprogrammaufruf. Dieser Aufruf wird jedoch nicht durch einen CALL-Befehl veranlaßt, sondern hardwaremäßig durch ein über den IRQ-Pin des Z80A kommendes Signal. Immer, wenn dieser Pin auf "low" (null Volt) liegt, wird nach Abarbeiten des laufenden Befehls ein, ab der Adresse $0038 stehendes Unterprogramm (Interrupt-Routine) ausgeführt. So ist es möglich, Prozesse wie z.B. die

Tastaturabfrage im Hintergrund laufen zu lassen, während ein Basic-Programm ausgeführt wird.

Die Interrupt-Steuerung wird durch Event-Blocks, Chains und die Befehle EVERY und AFTER auch für Basic-und Maschinen-Unterprogramme nutzbar (mehr darüber in den Abschnitten 1.3.5, 3.1.5, 3.4.4, 3.7.2, 3.8.7 und dem Kapitel 4.7). Die Adresse der Interrupt-Routine kann auch über das Interrupt-Register (I) und ein Byte generiert werden, das durch den Interrupt-auslösenden Baustein auf den Datenbus gelegte wird. Dieser Interrupt-Modus wird sinnvollerweise bei mehreren Interrupt-Quellen eingesetzt. Im Schneider CPC existiert jedoch nur eine solche Quelle: das Gate-Array. Die Adresse der Interrupt-Routine liegt hier durch Auswahl des Interrupt-Modus 1 (IM 1) stets bei $0038.

1.2 Videocontroller 6845 CRTC

1.2.1 Einführung

Die Abkürzung CRTC (*C*athode *R*ay *T*ube *C*ontroller, Anhang A2.1 und A2.2) bedeutet übersetzt: Steuerbaustein für Kathodenstrahlröhren (z.B. in einem Fernseher oder Monitor). Der 6845 hat im CPC die Aufgabe, die Adressen zum Auslesen des Video-RAMs (in der richtigen Reihenfolge) zu generieren. Das Video-Signal wird jedoch nicht vom 6845 erzeugt, sondern vom Gate Array 20 RA 043 (siehe nächstes Kapitel).

1.2.2 Die Pins des 6845

D0 bis D7: Durch diese Pins ist der 6845 an den Datenbus des Z80A angeschlossen.

Enable (E): Dieser Pin wird normalerweise an den Prozessortakt angeschlossen, um den Zeitpunkt der Übernahme der auf dem Datenbus liegenden Daten festzulegen. Im CPC ist hier ein aus -IORD und -IOWR gewonnenes Signal angeschlossen, das dem invertierten, vom Z80A kommenden, -IORQ entspricht.

Read/Write (R/-W): Mit diesem Pin wird festgelegt, ob Daten zum oder vom 6845 transferiert werden. Da der Z80A dieses Signal nicht liefert und die Entwickler des CPC die Schaltung einfach und kostengünstig halten wollten, ist dieser Pin mit dem Adreßbus (Pin A9) verbunden. Es wird also über die Adresse ausgewählt, ob gelesen oder geschrieben wird. In der Programmierung muß man daher darauf achten, nur auf solche Adressen zu schreiben, die auch wirklich dafür vorgesehen sind.

Chip Select (-CS): Dieser Pin sagt dem 6845, daß eines seiner Register vom Prozessor angesprochen werden soll. Er ist mit dem Pin A14 vom Adreßbus verbunden. Der 6845 wird also von allen I/O-Adressen angesprochen, deren 14. Bit gleich null ist. Wegen der auch bei anderen Bausteinen unvollständigen Adreßdecodierung sollten jedoch die übrigen Bits der Adresse, zumindest des Hi-Bytes, gleich eins sein.

Register Select (RS): Ob bei einem Zugriff auf den 6845 das Adreß- oder das Datenregister (siehe unten) angesprochen werden soll, wird mit diesem Pin festgelegt, der mit dem Adreßpin A9 verbunden ist.

Horizontal Synchronization (HSYNC): Am Ende jeder Rasterzeile wird ein Impuls auf diesem Pin ausgeben, um den Anfang der nächsten Zeile zu synchronisieren. Dieses Signal wird vom Gate Array zur Erzeugung des Videosignals weiterverarbeitet.

Vertical Synchronization (VSYNC): Dieser Pin entspricht dem HSYNC-Signal, nur ist er für die vertikale Synchronisation am Ende eines Bildaufbaus zuständig.

Display Enable (DISPEN): Durch diesen Pin teilt der 6845 CRTC dem Gate Array mit, wann Daten aus den Bildschirmspeicher ausgelesen werden müssen und wann die nicht beschreibbare Fläche des Bildschirms dargestellt wird.

Refresh Memory Adresses - (MA0 bis MA13): Der 6845 legt durch diese Pins fest, welche Adresse gerade ausgelesen werden soll.

Raster Adresses (RA0 bis RA4): Diese Pins dienen zur Adressierung eines Character-ROMs. Sie bestimmen die Rasterzeile innerhalb eines Zeichens, die gerade bearbeitet wird.

Cursor: Falls der Cursor auf dem Bildschirm hardwaremäßig dargestellt werden soll, werden die Daten aus dem Bildschirmspeicher vor Generierung des Videosignals mit diesem Signal verknüpft. Im Schneider CPC wird der Cursor jedoch softwaremäßig erzeugt.

Clock (CLK): Über dieses Signal werden alle Vorgänge im 6845 synchronisiert. Der CLK-Takt legt den Takt fest, mit dem die Zeichen auf dem Bildschirm dargestellt werden. Wenn DISPEN auf "high" ist, liegt bei MA0 der halbe CLK-Takt an, bei MA1 ein Viertel usw..

Light Pen Strobe (LPSTR): Wenn dieses Signal von "low" auf "high" wechselt, dann wird die gerade ausgelesene Bildschirmadresse zwischengespeichert, um eine Bestimmung der Bildschirm-Position eines angeschlossenen Lichtgriffels möglich zu machen.

Reset (-RES): Dieser Pin wird benutzt, um den 6845 CRTC in den Ausgangszustand zurückzusetzen. Ein Signal auf diesem Pin ist nur wirksam, wenn LPSTR auf "low" ist. Die internen Register behalten nach einem Reset ihren alten Wert.

1.2.3 Register des 6845

Der 6845 besitzt zur Speicherung der Parameter, die er für seine Aufgabe bentötigt, 19 interne Register. Auswählbar sind diese Register durch die Pins RS, -CS und R/-W. Im Schneider können sie über die I/O-Adressen

$BCxx, $BDxx und $BFxx angesprochen werden. Wie man sieht, ist das Lo-Byte der Adresse hierbei uninteressant, die Auswahl erfolgt allein über das Hi-Byte. Das bei $BCxx liegende Register dient als Adreß-Register. Hier wird die Nummer (von 0 bis 17) des Registers hineingeschrieben, auf das man als nächstes zugreifen möchte. Das Adreß-Register kann nicht ausgelesen werden. Nachdem eine Nummer ins Adreß-Register geschrieben wurde, kann man über die I/O-Adresse $BDxx das ausgewählte Daten-Register beschreiben, über die Adresse $BFxx dagegen aus ihm lesen. Durch diese Technik ist es möglich, 18 verschiedene Datenregister zu verwalten und trotzdem mit wenigen I/O- Adressen bzw. Auswahlpins (RS) auszukommen.

Die Bedeutung der einzelnen Daten-Register (In die Register kann nur geschrieben werden, wenn nicht anders angegeben):

0 **Horizontal Total Register:** In diesem Register steht die Zahl der Zeichen pro Zeile (minus 1) einschließlich der nicht beschreibbaren Randfläche. Dieses Register bestimmt damit die Anzahl der CLK-Takte, die vergehen, bis ein HSYNC-Signal (Horizontal Synchronization) ausgegeben wird.

1 **Horizontal Displayed Register:** In diesem Register steht die Zahl der tatsächlich angezeigten Zeichen pro Zeile.

2 **Horizontal Sync Position Register:** Hier ist die Position (in CLK-Takten) des HSYNC-Signals in einer Rasterzeile enthalten. Mit diesem Register läßt sich das ganze Bild in waagerechter Richtung verschieben.

3 **Horizontal Sync Width Register:** Dieses Register bestimmt die Länge des HSYNC-Impulses in CLK-Takten. Es werden nur die untersten 4 Bits verwendet.

4 **Vertical Total Register:** Durch dieses 7-Bit-Register wird die Zahl der Zeichenzeilen (minus 1) inklusive des nicht beschreibbaren Randes festgelegt. Mit Hilfe von Register 9 und der Frequenz des HSYNC-Signals (bestimmt durch Register 0) läßt sich die Frequenz des VSYNC-Signals (Vertical Synchronization) errechnen. Sie ist im CPC zu 50 Hz bzw. 60 Hz gewählt worden.

5 **Vertical Total Adjust Register:** Mit diesem 5-Bit-Register kann eine Feineinstellung (in Rasterzeilen) des Wertes aus Register 4 vorgenommen werden.

6 **Vertical Displayed Register:** Dieses 7-Bit-Register gibt die Zahl der tatsächlich angezeigten Zeichenzeilen an.

7 **Vertical Sync Position Register:** Die Position des VSYNC-Signals (in Zeichenpositionen) wird durch dieses 7-Bit-Register festgelegt. Mit diesem Register läßt sich das Bild in senkrechter Richtung verschieben.

8 **Interlace Mode Register:** Die unteren beiden Bits bestimmen, ob das Monitorbild im Interlace-Mode (Zeilensprung- Verfahren) aufgebaut wird.

9 **Maximum Scan Line Address Register:** In diesem 5-Bit-Register wird die Höhe eines Zeichens in Rasterzeilen festgelegt.

10/11 **Cursor Start Register, Cursor End Register:** Diese Register bestimmen Start- und End-Rasterzeile des hardwaremäßig erzeugten Cursors. Im Schneider CPC wird der Cursor jedoch softwaremäßig erzeugt.

12/13 **Start Address Register:** Mit diesem 14-Bit-Register kann die Startadresse des Bildschirmspeichers beliebig verschoben werden.

14/15 **Cursor Register:** Mit diesem 14-Bit-Register kann die Cursoradresse gesetzt oder gelesen werden.

16/17 **Light Pen Register:** Im diesem 14-Bit-Register wird die Bildschirmadresse bei einem aufgetretenen LPSTRB-Signal abgelegt und kann dann ausgelesen werden.

1.2.4 Aufbau und Abfrage des Video-RAMs

Der 6845 ist für die Erzeugung von Zeichen auf dem Bildschirm mit Hilfe eines Video-RAMs und eines Zeichen-Generators im ROM vorgesehen. Im CPC ist er (über Register 1, 6 und 9) auf eine Zeichenhöhe von acht Rasterzeilen bei 25 Zeichenzeilen zu je 40 Zeichen eingestellt, womit sich 25*8 = 200 Rasterzeilen ergeben. Dies widerspricht zunächst der maximalen Auflösung des CPC von 25 Zeilen zu je 80 Zeichen.

Da der 6845 weder ein Video-Signal erzeugt, noch das Video-RAM ausliest, wird die Breite der Zeichen auf dem Bildschirm (jedenfalls im Schneider Computer) nicht von ihm bestimmt, sondern vom Gate Array. Das Gate-Array liest bei jeder Adresse, die der 6845 liefert, zwei Bytes aus dem Speicher. Mit zwei Bytes lassen sich maximal 16 Punkte darstellen: Die

Breite eines Zeichens ist also maximal 16 Punkte, die Zahl der Punkte pro Zeile beträgt maximal 40*16 = 640. Damit bei jeder vom CRTC über die Pins MA0 bis MA13 gelieferten Adresse zwei Bytes aus dem Video-RAM ausgelesen werden können, müssen die Adressen mit zwei multipliziert werden, dies entspricht einem Verschieben um eine Position. Entsprechend sind die Adreß-Pins des 6845 mit dem RAM verbunden:

Adreß-Pin entsprechend Z80A-Pin	vom 6845 CRTC
A 0	CLK
A 1	MA 0
A 2	MA 1
A 3	MA 2
A 4	MA 3
A 5	MA 4
A 6	MA 5
A 7	MA 6
A 8	MA 7
A 9	MA 8
A 10	MA 9
A 11	RA 0
A 12	RA 1
A 13	RA 2
A 14	MA 12
A 15	MA 13

Aus obiger Tabelle läßt sich ablesen, daß vom CRTC die gesamten 64 KByte RAM des CPC 464/664 adressiert werden können. Im CPC 6128 hat das Gate Array mit Hilfe der CRTC-Adressen hingegen nicht auf die gesamten 128 KByte RAM, sondern nur auf die ersten 64 KByte Zugriff (Bank 0 bis 3, siehe Kapitel 1.3).

Es läßt sich eine weitere Besonderheit im CPC erkennen: Eine Matrix von 25 mal 40 Zeichen würde einen Bildschirm-Adreßraum von 1000 CRTC-Adressen (2000 Bytes) benötigen. Zusätzlich zu den dafür benötigten 10 Adreßbits MA 0 bis MA 9 sind noch die eigentlich für das Auslesen eines Character-ROMs gedachten Anschlüsse RA 0 bis RA 2 an den Adreßbus des RAMs angeschlossen. Die zur Darstellung eines Zeichen benötigten Punkt-Matrizen, die normalerweise aus einem ROM gelesen werden, kommen im CPC aus dem RAM. Das heißt, daß im Video-RAM des Schneider Computers keine Zeichencodes ähnlich der ASCII-Codes gespeichert werden müssen, sondern daß jeder einzelne Punkt eines Zeichens extra gesetzt werden muß. Das heißt aber auch, daß beliebige Graphiken erzeugt werden können. Hier löst sich auch der obige scheinbare Widerspruch auf: Bei 40*16 = 640 Punkten pro Zeile können nämlich 80 Zeichen pro Zeile dargestellt werden. Dem 6845 wird jedoch vorgetäuscht, zwei dieser

Zeichen wären zusammen ein doppelt so breites Zeichen, was in Speicherorganisation und Auflösung keinen Unterschied macht.

Durch diese Organisationsform ist der Bildschirmspeicher wie folgt aufgebaut: Die ersten 2000 Bytes enthalten die Bitmuster der obersten Rasterzeilen der Zeichenpositionen (80 Byte pro Zeile, 25 Zeichenzeilen), da bei den ersten 2000 Bytes A 11 bis A 13 und damit RA 0 bis RA 2 auf Null liegen (durch RA 0 bis RA 2 wird die angesprochene Rasterzeile innerhalb eines Zeichens ausgewählt). Wenn man die ersten 2000 Bytes des Bildschirms mit einem Bitmuster füllt, das die Pixel weiß erscheinen läßt, so erscheinen auf dem Bildschirm weiße Punktzeilen im Abstand von je acht Rasterzeilen, da eine Zeichenposition ja acht Rasterzeilen hoch ist. Nach diesen 2000 Bytes folgen 48 unbenutzte Bytes. Bei den folgenden 2000 Bytes ist A 11 und damit RA 0 auf Eins, so daß nicht mehr die oberste, sondern die zweitoberste Rasterzeile in jedem Zeichen angesprochen wird. Diese 2000 Bytes wiederholen sich abwechselnd mit 48 unbenutzten Bytes für jede Rasterzeile im Zeichen, also insgesamt acht Mal. Die Größe des Bildschirmspeichers beträgt somit (2000+48)*8 = 16384 Byte.

Über die Register 12 und 13 des CRTC läßt sich die Startadresse des Bildschirmspeichers einstellen. Die Pins MA13 und MA12 sind mit den RAM-Adreßpins A14 und A15 verbunden. Durch die entsprechenden Bits in Register 12 läßt sich der Bildschirmspeicher daher in 16-KByte-Blöcken verschieben. Im CPC sind diese Bits normalerweise gesetzt, der Bildschirmspeicher liegt also ab $C000. Dieser Bildschirmstart ist zusätzlich in der Systemvariablen SCR BASE gespeichert. Innerhalb eines 16-KByte-Blocks kann die Startadresse über die 10 unteren Bits des 14-Bit-Registers 12/13, die den Pins MA0 bis MA9 entsprechen, noch einmal genauer eingestellt werden. Da pro CRTC-Adresse jedoch zwei Bytes ausgelesen werden, kann diese in der Systemvariable SCR OFFSET gespeicherten Startadresse auch nur in Zwei-Byte-Schritten eingestellt werden.

Wichtig ist nun, was geschieht, wenn SCR BASE auf $C000 zeigt und SCR OFFSET beispielsweise angibt, daß der Bildschirm erst ab $C100 beginnen soll. Man könnte vermuten, daß die letzten $100 Bytes des Bildschirmspeichers dann von $0000 bis $0100 liegen. Dies ist aber nicht der Fall. Die beiden obersten Bits der Bildschirmadresse, A14 und A15, liegen nämlich immer fest. Ein Übertrag von SCR OFFSET (entspricht MA0 bis MA9) wird vom CRTC durch den Pin MA10 angezeigt. MA10 und MA11 sind jedoch im Schneider CPC gar nicht verbunden, beeinflussen also die an den RAMs anliegenden Adressen auch nicht. Wegen dieser fehlenden Übertrags-Möglichkeit ist der Bildschirmspeicher in acht separate 2-KByte-Blöcke aufgeteilt (beginnend z. B. bei $C000, $C800, $D000 usw.), von

denen der erste die jeweils oberste Rasterzeile der Zeichenpositionen, der zweite die zweitoberste Zeile usw. darstellt.

SCR OFFSET gibt nun genau genommen für jeden dieser acht 2-KByte-Blöcke den Bildschirmstart an. Wenn SCR OFFSET den Wert $100 besitzt, so liegt die Information für die Zeichen-Position oben links bei $C100, $C900, $D100 usw., je nach Rasterzeile. Auch wenn der erste 2-KByte-Block jetzt bei $C100 zu beginnen scheint, so belegt er dennoch den Bereich von $C000 bis $C7FF. Den ersten 1792 ($700) Zeichenpositionen auf dem Bildschirm entspricht der Bereich von $C100 bis $C7FF. Die restlichen 2000-1792=208 (=$D0) Positionen werden mit den Bytes $C000 bis $C0CF dargestellt. Die 48 Bytes von $C0D0 bis $C0FF bleiben frei. Die oben angeführte Aufteilung in jeweils 2000 benutzte und 48 unbenutzte Bytes ist also nur bei SCR OFFSET = 0 korrekt, in anderen Fällen liegen die 48 unbenutzten Bytes mitten im 2-KByte-Block, und zwar nach der letzten und vor der ersten dargestellten Zeichenposition.

Aus der Einteilung in separate 2-KByte-Blöcke folgt, daß für SCR OFFSET nur Werte kleiner $800 gewählt werden können. SCR OFFSET stellt also einen 10-Bit-Wert dar. Durch Verändern von SCR OFFSET kann der Bildschirm relativ schnell hardwaremäßig gescrollt werden. Es brauchen beim Scrollen also nicht zeitaufwendig acht 2-KByte-Blöcke in sich verschoben zu werden. Dieser Vorteil hat jedoch auch einen Nachteil: Der Sprung von z.B. $C7FF bei einer Zeichenposition zu $C000 bei der nächsten muß gesondert berücksichtigt werden. Dieser Sprung kann (bei einem Start ab $C7F0 beispielsweise) auch mitten in einer Textzeile auftreten. Überträge zu den RA-Bits (ab Bit 10, entspricht $800) müssen also beim fortlaufenden Erhöhen der Bildschirmadresse wieder ausgeglichen werden.

1.3 Gate Array 20 RA 043

Das Gate Array 20 RA 043 ist ein Baustein, der viele verschiedene Bausteine ersetzt. Es kann vom Hersteller entsprechend der Anwendung innerhalb gewisser Grenzen programmiert werden. Über den genauen Aufbau ist schwer etwas herauszufinden, es sind auch keine Datenblätter oder Beschreibungen erhältlich. Wir können diesen Baustein daher nur durch seinen Anschluß und seine Programmierung im CPC beschreiben.

Das Gate Array hat im Schneider CPC verschiedene Aufgaben. Es erzeugt den Systemtakt und andere Takte, generiert das Videosignal und stellt einen Interrupt-Takt zur Verfügung. Weiterhin steuert es Speicherzugriffe auf RAM und ROM.

1.3.1 Erzeugung der Takte

Das Gate Array wird über den Pin XTAL mit einem 16-MHz-Takt versorgt, den ein Quarz erzeugt. Dieser Takt dient als Basis für die vom Gate Array erzeugten Takte. Einen 4-MHz-Takt gibt das Gate Array als Systemtakt für den Z80A-Prozessor aus. Zur Steuerung der RAM-Zugriffe für das Videosignal und vom Prozessor dient der Takt CPU ADDR. Er hat eine Frequenz von 1 MHz und dient zusätzlich noch als Zeitbasis für den Sound-Chip AY-3-8912. Ein weiterer 1-MHz-Takt ist der Takt für den CRTC, der am Pin CCLK anliegt.

1.3.2 Pinbelegung

Im folgenden beschreiben wir die einzelnen Pins des Gate Arrays im CPC 464 (siehe Anhang A3.1). Die GAs von CPC 664 und CPC 6128 haben noch zusätzliche Pins, die uns bei Drucklegung nicht bekannt waren.

XTAL: Hier liegt der durch einen 16-MHz-Quarz erzeugte Takt an, der Grundlage für alle vom Gate Array erzeugten Takte ist.

PHI: Hier liegt der 4-MHz-Systemtakt für den Prozessor an.

VSYNC, HSYNC, DISPEN: Diese Signale kommen vom 6845 CRTC (siehe Kapitel 1.2). Sie sind zur Erzeugung des Videosignals durch das Gate Array notwendig.

R, G, B, -SYNC: An diesen Pins wird das RGB-Videosignal ausgegeben. Es wird außerhalb des Gate Array für den Betrieb eines monochromen Monitors weiter aufbereitet.

-CPU ADDR: Dieses 1-MHz-Signal bestimmt, wer auf das RAM zugreifen darf. Wenn es auf "high" liegt, kann das Gate Array mit Hilfe der Adressen vom CRTC das Video-RAM auslesen, sonst hat die Z80A-CPU Zugriff auf das RAM.

-RAS, -CAS: Da die Adressen in zwei Hälften an die RAMs übergeben werden, sind zwei Signale nötig, die bestimmen, *welche* Hälfte gerade übergeben wird. Bei einem RAM-Zugriff wird zunächst die eine Adreßhälfte zusammen mit dem -RAS-Signal an die RAMs ausgeben und dann die andere Hälfte zusammen mit dem -CAS-Signal. Über diese Signale wird also das RAM überhaupt erst ausgewählt.

-CAS ADDR: Mit diesem Signal wird zwischen den beiden Adreßbus-Hälften umgeschaltet, damit zusammen mit -CAS bzw. -RAS jeweils die richtige Hälfte an den RAMs anliegt.

-MWE: Den RAMs wird über diesen Pin mitgeteilt, ob gelesen oder geschrieben werden soll.

-RAMRD: Dieses Signal veranlaßt einen anderen Chip, die vom RAM kommenden Daten zu sperren, solange es "high" ist. Auf diese Weise ist es möglich, auch das ROM auszulesen.

-ROMEN: Über diesen Pin wird das ROM für Lesezugriffe selektiert.

READY: Für den Aufbau des Video-Bildes ist es unerläßlich, daß das Gate Array beim Auslesen des RAMs Vorrang vor dem Prozessor hat. Deswegen kann der Prozessor über diesen Pin, der mit dem WAIT-Eingang des Z80A verbunden ist, vom Gate Array in einen Wartezustand versetzt werden. Für den Prozessor schon aus dem RAM bereitstehende Daten, werden für die Dauer dieses Signals in einem externen Chip zwischengespeichert.

-244EN: Über diesen Pin kann das Gate Array über einen anderen Chip auf den Datenbus des Prozessors zugreifen, an den es nicht direkt angeschlossen ist. Das ist nötig, wenn der Prozessor in ein internes Register des Gate Array schreiben will.

MA0/CCLK: Dieser Pin liefert einen 1-MHz-Takt zur Steuerung des 6845 CRTC. Außerdem dient er als Adreßbit beim Auslesen des Video-RAMs.

D0 bis D7: Über diese Pins ist das Gate Array an den Datenbus der RAMs sowie an einen externen Chip angeschlossen, der über das Signal -244EN eine Verbindung zum Prozessor-Datenbus herstellen kann.

A14 und A15: Diese Pins sind mit den entsprechenden Pins des Prozessor-Adreßbusses verbunden. Sie ermöglichen das RAM/ROM-Banking und dienen zur Decodierung der Adresse des internen Gate-Array-Registers .

-RESET: Über diesen Pin kann das Gate-Array in den Ausgangszustand zurückgesetzt werden.

-RD: Dieses vom Prozessor kommende Signal zeigt an, daß der Prozessor einen Lesezugriff ausführen möchte. Das Gate Array generiert hieraus unter anderem -ROMEN und -RAMRD.

-MREQ: Der Prozessor zeigt durch einen Low-Pegel einen Speicherzugriff auf seinem -MREQ-Pin an. Das Gate Array benutzt dieses Signal, um die Signale -RAMRD und -ROMEN zu generieren.

-IORQ: Über diesen Pin erfährt das Gate Array, daß ein internes Register angesprochen wird.

-M1: Wenn der Z80A einen Interrupt bearbeitet, geht der -M1-Pin des Prozessors zusammen mit dem -IORQ-Pin auf "low". Dies dient eigentlich dazu, um das interrupt-auslösende Gerät zu veranlassen, ein Sprungvektor-Byte auf den Datenbus zu legen. Das Gate Array benutzt diesen Pin nur, um den Interrupt zu erkennen.

-INTERRUPT: Über diesen Pin gibt das Gate Array Interrupt-Impulse an den Z80A aus. Ein Impuls dauert so lange, bis das Gate Array die Behandlung des Interrupts durch den Z80A über -IORQ und -M1 erkennt.

1.3.3 Die Register des Gate Array

Das Gate Array ist über die I/O-Adresse $7Fxx ansprechbar (das Lo-Byte der Adresse ist nicht relevant). Diese Adresse kann nur beschrieben werden. Die beiden obersten Bits eines an diese Adresse geschriebenen Bytes wählen die Nummer des Registers aus. So kann mit nur einer Adresse auf vier 6-Bit-Register zugegriffen werden.

Register 0 (b7=0, b6=0) - Farb-Adreß-Register
Register 1 (b7=0, b6=1) - Farbwert-Daten-Register
Register 2 (b7=1, b6=0)
b0/b1 : Auswahl des MODE
00 : Mode 0
01 : Mode 1
10 : Mode 2
11 : wie Mode 0
b2 : 0 = ROM $0000 bis $3FFF einschalten
b3 : 0 = ROM $C000 bis $FFFF einschalten
b4 : 1 = Interrupt-Zähler löschen
b5 : Funktion nicht bekannt

Register 3 (b7=1, b6=1) - RAM-Bank-Register im CPC 6128
b0-b2 : Auswahl der RAM-Konfiguration (Abb. 1.2)
b3-b5 : Funktion nicht bekannt

1.3.4 Erzeugung des Videosignals

Das Gate Array erzeugt mit Hilfe der Synchronisations-Signale des CRTC und der Daten aus dem Video-RAM - zu denen der CRTC die Adressen liefert - das RGB-Video-Signal. Wie schon im Kapitel über den CRTC beschrieben, liest das Gate Array zwei Bytes pro CRTC-Adresse. Dazu sind die CRTC-Adreßbits um ein Bit verschoben. Als unterstes Adreßbit A0 wird das Signal CCLK des Gate Array verwendet, das die Zeitbasis des CRTC bildet. Dies ist möglich, da pro CRTC-Ardesse CCLK einmal auf "low" und einmal auf "high" liegt. CCLK hat also die doppelte Frequenz des untersten CRTC-Adreßpins MA0.

Die Punkte, die das Gate Array entsprechend den Daten aus dem Video-RAM darstellt, können 32 verschiedene Farben haben. (Daß im Handbuch von 27 Farben gesprochen wird, liegt daran, daß fünf Farben jeweils fünf anderen Farben entsprechen.) Das Gate Array besitzt (zusätzlich zu den oben angesprochenen) 17 Register, in die Farbwerte gespeichert werden können. Diese Register werden im folgenden Farbwert-Register genannt, um Verwechslungen mit den oben aufgeführten Registern 0 bis 3 zu vermeiden. Auf welche Weise werden diese 17 Farbwert-Register nun verwaltet?

Register 0 ist ein Adreß-Register. Es bestimmt, welches Farbwert-Register als Register 1 mit einem Farbwert beschrieben werden kann. Diese Zuordnung von Farbwerten zu Farbwert-Registern entspricht den Basic-Befehlen INK und BORDER. Die Farbstiftnummer von 0 bis 15 beim Ink-Befehl entspricht den Farbwert-Register-Nummern im Adreß-Register. Das Farbwert-Register 16 enthält den BORDER-Farbwert. Die in die Farbwert-Register eingetragenen Farbwerte entsprechen allerdings nicht den bei INK und BORDER angegebenen Farben. Die Umcodierung dieser Farbwerte geschieht nach folgender Tabelle:

INK/ BORDER	Farbwert-Register	INK/ BORDER	Farbwert-Register
0	20	16	7
1	4	17	15
2	21	18	18
3	28	19	2
4	24	20	19
5	29	21	26
6	12	22	25
7	5	23	27
8	13	24	10
9	22	25	3
10	6	26	11
11	23	27	1
12	30	28	8
13	0	29	9
14	31	30	16
15	14	31	17

Es mag verwunderlich erscheinen, daß die bei INK und BORDER angegebenen Farben erst nach obiger Tabelle umgewandelt werden, bevor sie in das Farbwert-Register mit der Nummer des Farbstifts geschrieben werden. Die Umwandlung beläßt die Werte im gleichen Bereich: von 0 bis 31. Sie ist aber dennoch sinnvoll, denn die 5 nicht benutzten Farben, die 5 anderen entsprechen, liegen so im Bereich von 27 bis 31 und sind daher leichter zu behalten.

Wieviele Farben, außer der Farbe des Bildschirm-Rahmens, gleichzeitig auf dem Bildschirm dargestellt werden können, hängt vom Bildschirm-Modus ab, der mit dem MODE-Befehl eingeschaltet wird. Die Nummer des Modus wird dem Gate Array über die Bits b0 und b1 in Register 2 mitgeteilt. Vom eingeschalteten Modus hängt es auch ab, wieviele Punkte pro ausgelesenem Byte das Gate Array auf dem Bildschirm darstellt.

Modus	0	1	2
Reg. 2 b0	0	1	0
Reg. 2 b1	0	0	1
Zahl der Farben ohne Rand	16	4	2
Bits pro Punkt	4	2	1
Punkte pro Byte	2	4	8
Punkte pro CRTC-Adresse	4	8	16

Wie werden nun die einzelnen Bytes aus dem Video-RAM zu Punkten auf dem Bildschirm gewandelt?

Für Mode 2 ist die Darstellung relativ einfach. Jeder Punkt entspricht genau einem Bit. Der ganz links liegende Punkt entspricht dem höchsten

Bit im Byte. Ist ein Bit gesetzt, so wird der Punkt mit der Farbe aus Farbwert-Register 1 dargestellt, sonst in der Farbe aus Farbwert-Register 0.

Bei Mode 1 entspricht ein Punkt zwei Bits, jeweils einem Bit aus dem unteren und dem oberen Nibble (Halbbyte). Mit diesen beiden Bits kann aus vier Farben, den Farben aus Farbwertregister 0 bis 3, ausgewählt werden.

Bei Mode 0 wird die Farbe eines Punktes durch vier Bits bestimmt. Die geraden Bits im Byte bestimmen die Farbe des rechten, die ungeraden die des linken Punktes.

Zur Verdeutlichung nun eine Tabelle. b0 bis b7 sind die einzelnen Bits eines Bytes im Video-RAM. Es werden die Punkte A bis H dargestellt, wobei A der ganz links stehende Punkt ist. (Bei Mode 0 bzw. 1 werden nur die Punkte A bis B bzw. A bis D dargestellt.) Ein C2 in der Tabelle bedeutet beispielsweise: Bit 2 der Nummer des Farbwert-Registers derjenigen Farbe, in der Punkt C dargestellt wird.

Bit	b7	b6	b5	b4	b3	b2	b1	b0
Mode 2	A0	B0	C0	D0	E0	F0	G0	H0
Mode 1	A0	B0	C0	D0	A1	B1	C1	D1
Mode 0	A0	B0	A2	B2	A1	B1	A3	B3

1.3.5 Erzeugung des Interrupt-Signals

Das Interruptsignal für den Z80A wird durch Teilung des HSYNC-Signals durch 52 erreicht. Da das HSYNC-Signal eine Frequenz von 15625 Hz (bei 50 Hz Bildwiederholfrequenz) besitzt, beträgt die Frequenz des Interruptsignals ca. 300 Hz. Der HSYNC-Zähler für das Interruptsignal kann durch Setzen von Bit 4 in Register 2 zurückgesetzt werden. Das Interruptsignal liegt solange an, bis der Z80A auf den Interrupt reagiert hat. Damit wird erreicht, daß eine möglichst große Anzahl von Interrupts auch bedient werden kann. Falls eine zusätzliche Interruptquelle über den Expansion-Bus angeschlossen wird, muß das Interruptsignal solange anliegen, bis es softwaremäßig zurückgesetzt wird. In der Interrupt-Routine wird nämlich für kurze Zeit der Interrupt mittels EI wieder freigegeben, um zu prüfen, ob eine externe Interruptquelle Auslöser des Interrupts war. Wenn bei dieser Freigabe kein neuer Interrupt ausglöst wird, so hat das Gate Array den Interrupt ausgelöst.

1.3.6 Speicherverwaltung im CPC 464/664

Der Schneider CPC 464/664 hat neben 64-KByte RAM auch noch 32 KByte ROM zu verwalten. Deshalb müssen einige Adressen des 64KByte-Adreßraums des Z80A mehrfach belegt werden. So kann unter den Adressen $0000 bis $3FFF sowohl ROM als auch RAM angesprochen werden. Die Auswahl erfolgt durch Bit 2 von Register 2. Ebenso ist der Bereich von $C000 bis $FFFF mehrfach belegt, ausgewählt werden kann hier durch Bit 3 von Register 2. Diese Auswahl gilt aber nur für Lesezugriffe. Bei Schreibzugriffen wird immer das RAM angesprochen.

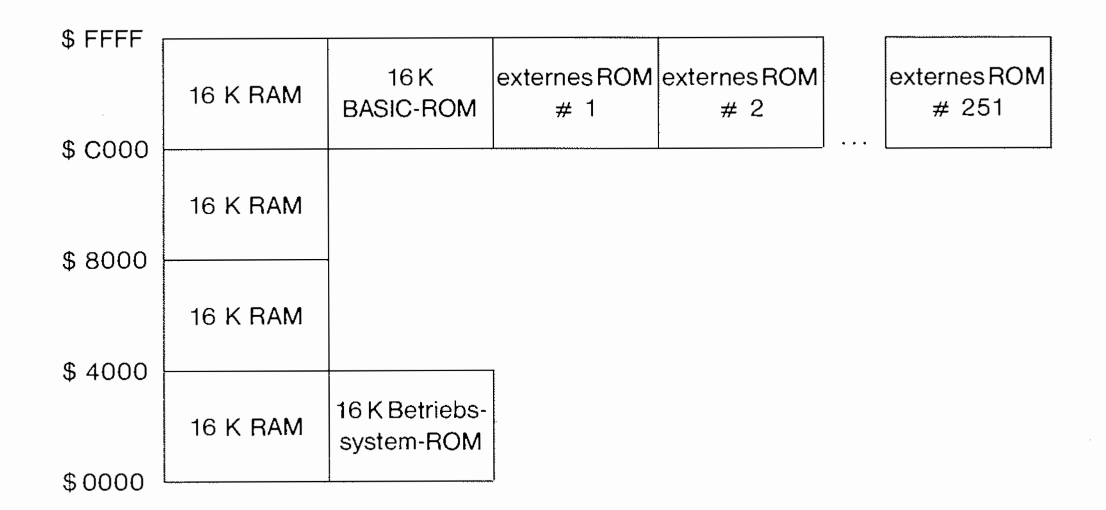


Abbildung 1.1: *Speicheraufteilung im CPC*

Im Bereich $C000 bis $FFFF können außer dem eingebauten Basic-ROM auch noch bis zu 251 externe 16-KByte-Erweiterungs-ROMs angesprochen werden. Die dazu nötige Auswahllogik wird an den CPC 464 erst bei Bedarf angeschlossen, wie z.B. beim Anschluß einer Diskettenstation mit dem dazu nötigen DOS-ROM. Im CPC 664 ist sie bereits enthalten. Die Nummer des auszuwählenden ROMs wird nicht dem Gate Array, sondern eben jener Auswahllogik (I/O-Adresse $DFxx) übergeben. Eine Null bezeichnet dabei das eingebaute Basic-ROM, die Werte 1 bis $FB (dezimal 251) wählen an dessen Stelle externe ROMs aus (eine 7 z.B. das DOS-ROM). Zur Veranschaulichung des RAM-/ROM-Bankings dient Abbildung 1.1.

1.3.7 Speicherverwaltung im CPC 6128

Im Gegensatz zu seinen beiden Vorgängern, muß der CPC 6128 128 KByte RAM und 32 KByte ROM mit Hilfe von nur 64 KByte Adreßraum verwalten. Die 128 KByte RAM lassen sich in acht 16-KByte-Bereiche (sogenannte Banks) einteilen. Wir nennen sie der Einfachheit halber Bank 0 bis Bank 7. Im vorhandenen Adreßraum können maximal vier Banks gleichzeitig angesprochen werden. Um diese auszuwählen und den Adreßbereich, in dem sie liegen, zu bestimmen, dient das Register 3 des Gate Arrays. Mit den drei benutzten Bits dieses Registers sind acht Konfigurationen möglich, die in Abbildung 1.2 dargestellt werden. Nach dem Einschalten des CPC 6128 ist Konfiguration 0 selektiert. Die Auswahl der RAM-Konfiguration hat keinen Einfluß auf das Video-RAM: es liegt (je nach SCR OFFSET) stets auf Bank 0, 1, 2 oder 3. Ebenso können die ROMs unabhängig von der RAM-Konfiguration, wie in Abschnitt 1.3.6 beschrieben, angesprochen werden.

	0	1	2	3	4	5	6	7
$FFFF – $C000	Bank 3	Bank 7	Bank 7	Bank 7	Bank 3	Bank 3	Bank 3	Bank 3
$C000 – $8000	Bank 2	Bank 2	Bank 6	Bank 2	Bank 2	Bank 2	Bank 2	Bank 2
$8000 – $4000	Bank 1	Bank 1	Bank 5	frei	Bank 4	Bank 5	Bank 6	Bank 7
$4000 – $0000	Bank 0	Bank 0	Bank 4	Bank 0	Bank 0	Bank 0	Bank 0	Bank 0

Abbildung 1.2: *Inhalt von Register 3 des Gate Array*

1.4 Die 8255-PIO

1.4.1 Allgemeines

Die 8255 PIO (Programmable Input/Output, Anhang A4.2) gehört zur Klasse der Ein-/Ausgabechips. Der 8255 ist ein sehr allgemein verwendbarer Chip (ganz im Gegensatz zu Floppy Controller Chips, DMA Controller, CRTCs etc.), der nicht auf eine spezielle Art der I/O zugeschnitten ist. Ein solcher Baustein verfügt über Leitungen, deren (elektrischer) Zustand er abfragen (input) bzw. beeinflussen (output) kann. Dies geschieht natürlich auf digitaler Basis, d.h. es können nur jeweils zwei Zustände pro Leitung angenommen werden. Das gilt bei jeder Art von I/O-Operation. Weiterhin sind die I/O-Leitungen zu Bündeln von jeweils acht zusammengefaßt, um so mit einem Zugriff des Prozessors möglichst viele Leitungen ansprechen zu können (acht I/O-Leitungen entprechen der Breite des Datenbusses). Im Folgenden werden wir uns damit befassen, wie der 8255 im Schneider Computer (es gibt darin nur einen davon) durch den Benutzer programmiert werden kann.

1.4.2 Die Programmierung der 8255-PIO

Die Programmierung des 8255 geschieht über sieben interne Register, auf die der Programmierer zugreifen kann. Sie teilen sich im wesentlichen in zwei Gruppen: die 6 Datenregister und das Kontroll-Register.

Der 8255 besitzt 24 I/O-Leitungen, die zu drei Bündeln mit je acht Leitungen zusammengefaßt sind, den sogenannten I/O-Ports. Die Ports des 8255 tragen die Namen Port A, B, C. Jedem Port sind zwei Register zugeordnet, eines für die Ausgabe-Daten (der Prozessor kann durch einen Schreibzugriff auf diese Register die Zustände der Ausgabeleitungen beeinflussen), und eines für die Eingabe-Daten. (Hier können die Zustände der peripheren Leitungen vom Prozessor gelesen werden.) Normalerweise sind I/O-Bausteine wie der 8255 derart beschaffen, daß der Benutzer für jeden Port (bei anderen Bausteinen manchmal sogar für jede Leitung einzeln) bestimmen kann, ob die ihm zugeordneten Leitungen die Daten im Ausgabe-Register repräsentieren sollen (man sagt, der Port sei auf Ausgabe geschaltet), oder ob die Daten im Eingabe-Register die Zustände der entsprechenden Pins wiedergeben sollen (der Port also auf Eingabe geschaltet ist).

Solcherlei Informationen, die den augenblicklichen Status des Chips widerspiegeln bzw. der Kontrolle und der Koordination der Abläufe in der PIO dienen, werden im "Kontroll-Register" abgelegt. Da ein solches Kontroll-

Register jedoch nur maximal 8 Bits umfassen kann (im 8255 sind es sogar nur 7 Bits, da ein Bit für die Implementierung eines besonderen Features benötigt wird), konnten nicht sämtliche Informationen über alle drei Ports darin untergebracht werden. Man teilte deshalb die 24 I/O-Pins des 8255 in zwei Gruppen zu je 12 Leitungen, die dann jeweils gemeinsam von je einem Satz Bits im Kontroll-Register gesteuert werden. Die erste Gruppe, Gruppe A, umfaßt den Port A und die oberen 4 Bits von Port C. Die zweite Gruppe, Gruppe B, setzt sich aus dem Port B und der unteren Hälfte von Port C zusammen.

Das Kontroll-Register hat nun die folgende Gestalt:

```
b7-------b6--b5--b4---b3-------b2--b1---b0---

SF=0 .. MS1 MS0 A-IO Ch-IO .. MS0 B-IO Cl-IO
SF=1    x   x   x    N2       N1  N0   B
```

Das SF-Bit ist jenes Bit, das zum Einführen dieses besonderen Features dient, das wir oben bereits erwähnt hatten. Die restlichen Bits haben überhaupt nur die oben mnemonisch angegebene Bedeutung, wenn dieses Bit = 0 ist. Ist SF = 1, so kann ein beliebiges Bit im Ausgabe-Register von Port C gesetzt bzw. gelöscht werden, indem in N2, N1, N0 die Bitnummer und in B der gewünschte Zustand des Bits geschrieben wird. Um einzelne Bits in Port C zu beeinflussen, muß man also nicht erst Port C lesen, den Wert mit einer entsprechenden Maske verknüpfen und dann wieder zurückspeichern. Der Benutzer kann dagegen sehr schnell und bequem auf ein einzelnes Bit zugreifen.

Wenn das SF-Bit = 0 ist, dann hat das Kontroll-Register seine "normale" Funktion. Es kontrolliert die internen Abläufe im 8255, und zwar für jede der beiden oben angegebenen Gruppen einzeln. Die Bits b3 bis b6 dienen zur Kontrolle der Gruppe A (Port A und Port C, upper). Die beiden MS-Bits bestimmen den "Mode", in dem die Gruppe benutzt wird. Da im Schneider Computer lediglich der Mode 0 (bei dem beide Bits gelöscht sind) Verwendung findet, werden wir uns auch in unserer Besprechung auf diesen Modus beschränken. Zu allen anderen Modi (für die Gruppe A die Modi 1 und 2, für die Gruppe B nur noch Mode 1) sei nur kurz gesagt, daß sie vor allem dem Handshaking-Betrieb mit anderen Geräten dienen. Das Bit A-IO gibt an, ob der Port A auf Eingabe oder auf Ausgabe geschaltet werden soll. Ebenso gibt Ch-IO an, ob die obere Hälfte von Port C (also der Teil, der der Gruppe A zugeordnet wird) eine Eingabe- oder eine Ausgabeeinheit ist. Eine Null in einem dieser Datenrichtungs-Bits (so die Bezeichnung für derartige Kontroll-Bits, die ja die Richtung der Daten angeben, die über einen Port laufen) bedeutet, daß die entsprechenden Leitungen Ausgänge sind. Eine Eins macht sie zu Eingängen. (Dies ist

eigentlich ungewöhnlich, da vergleichbare Chips wie z.B. 6520/22/26, 6820/21 eine Null als Eingabe interpretieren. Bei einem Reset müssen alle Leitungen zunächst einmal wieder als Eingänge geschaltet werden. In den anderen Chips geschieht dies durch ein Löschen aller Register automatisch. Beim 8255 werden natürlich auch alle Ports als Eingänge gesetzt, und zwar, indem in das Kontroll-Register die entsprechenden Einsen gesetzt werden.)

Dies alles gilt sinngemäß natürlich ebenso für die Bits der Gruppe B (b0 bis b2). Auch hier wird nicht weiter auf den anderen Modus, in den diese Gruppe versetzt werden kann, eingegangen werden. Auch hier gibt es zwei Datenrichtungsbits: eines für den gesamten Port B, das andere für den unteren Teil des Ports C, der ja der Gruppe B zugeordnet ist. Auch in diesen Datenrichtungsbits meint eine Eins einen Eingang, eine Null schaltet die entsprechenden Leitungen als Ausgänge.

1.4.3 Der Zugriff auf die 8255-Register

Nachdem wir nun die Bedeutung der einzelnen Register im 8255 kennengelernt haben, wollen wir uns eben noch anschauen, wie man als Programmierer auf diese Register zugreifen kann. Die beiden Register eines Ports, die jeweils für die Ein- bzw. Ausgabe vorgesehen sind werden nicht voneinander getrennt sondern gemeinsam als "Port" angesprochen. Das jeweils richtige Register wird dann durch die interne Logik bestimmt.

Der 8255 besitzt zwei Pins, A0 und A1, durch die die Auswahl eines Registers unter den vier möglichen stattfindet (Kontroll-Register zuzüglich der drei Ports). Die Auswahl geschieht nach folgender Tabelle:

A1	A0	Adresse	ausgewählte Einheit
0	0	$F4xx	Port A
0	1	$F5xx	Port B
1	0	$F6xx	Port C
1	1	$F7xx	Kontroll-Register

Zu dieser Tabelle sei noch gesagt, daß auf die drei Ports natürlich Lese- und Schreibzugriffe erlaubt sind, und daß in das Kontroll-Register jedoch nur geschrieben werden kann. Das Lesen aus dem Kontroll-Register ist nicht zugelassen.

Die Werte in der Spalte "Adresse" sind die I/O-Adresse des jeweiligen PIO-Registers. Diese ergibt sich aus der Tatsache, daß A1 mit dem Adreßbusbit b9, A0 mit der Adreßleitung b8 und das "low"-aktive, zur Chip-Auswahl benötigte, -CS-Signal der PIO mit b11 des Adreßbusses verbunden ist. Alle anderen Leitungen müssen auf "high" liegen. Ein OUT &F500,&2F von

Basic aus schreibt so z.B. den angegebenen Wert ($2F) in das Ausgaberegister des Ports B.

1.4.4 Die Anwendung des 8255 im Schneider-Computer

Wie schon erwähnt, existiert im CPC nur ein einziger 8255. Im Folgenden wollen wir ein wenig näher auf die Belegung der einzelnen Port-Bits eingehen.

1.4.4.1 Die Abfrage der Tastatur

Die Tastatur des Schneider Computers ist in Form einer Matrix realisiert, die über neun Eingänge und acht Ausgänge verfügt. Daraus ergeben sich genau 9 * 8 = 72 Kreuzungspunkte. An jedem dieser Kreuzungspunkte ist genau eine Taste plaziert, wobei die beiden Shift-Tasten nicht getrennt, sondern parallel geschaltet sind. Hinzu kommt die DEL-Taste, die alleine einen zehnten Eingang belegt, was zusammen mit der doppelten Shift-Taste 74 Tasten ausmacht. (Die Aufstellung der Tastaturmatrix finden Sie im Anhang A4.4.)

Die Abfrage der Tastatur geschieht nun derart, daß jeweils einer der insgesamt zehn Eingänge (man spricht - das Bild der Matrix vor Augen - auch von einer Zeile) jeweils auf "low" gelegt wird und der Prozessor sich die sogenannte Rückmeldung abholt, d.h. er schaut nach, über welche Kreuzungspunkte sich dieser low-Impuls, den er in der Zeile anlegte, in die jeweilige Spalte übertragen hat. Dort muß dann ein geschlossener elektrischer Kontakt vorgelegen haben, das eine gedrückte Taste symbolisiert. Mit diesem Verfahren kann die CPU also eine Taste als einen Kreuzungspunkt in der Matrix bestimmen.

Der 8255 spielt bei der Tastaturabfrage natürlich die Rolle der Schnittstelle zwischen dem Computer und der Tastatur. Der Prozessor gibt über ihn die Nummer der Zeile aus, die er auf "low" legen möchte. Ein externer Demultiplexer legt dann die entsprechende Leitung auf null Volt. Anschließend liest der Prozessor ebenfalls über die PIO den Zustand der acht Kreuzungspunkte auf dieser Zeile in ein Byte hinein. Das Ausgeben der Zeilennummer geschieht in den unteren vier Bits des Ports C, das Einlesen der Rückmeldung wird über den Port A abgewickelt. (Dazu sei noch gesagt, daß zwischen den Spaltenausgängen der Tastatur und der PIO noch der Sound-Chip geschaltet ist. Dieser kann jedoch derart programmiert werden, daß er als einfacher I/O-Chip dient, d.h. er leitet die Rückmeldungen von der Tastatur einfach an die PIO weiter. Wir werden darauf noch näher im Kapitel über den PSG eingehen.)

Zur Tastaturabfrage sei noch angefügt, daß auch die Joysticks (inklusive Feuerknöpfe) mit diesem Verfahren ausgewählt und decodiert werden. Sie sind quasi in die Tastaturmatrix mit eingefügt.

1.4.4.2 Die Ausgabe von Sound

Obschon die Ausgabe von Sound im wesentlichen vom PSG (dem programmierbaren Sound-Generator) vollzogen wird, ist der 8255 dennoch in gewisser Weise daran beteiligt. Er dient der Zentraleinheit als Verbindung zum Sound-Chip, d.h. der PSG wird nur indirekt, über den 8255, programmiert. Die PIO versorgt der PSG zu diesem Zweck mit zwei Kontrollsignalen und acht Datenbits. Die acht Datenbits kommen aus dem Port A, sind also dieselben, mit denen auch die Rückmeldung von der Tastatur gelesen wird. Um die Datenrichtung festzulegen bzw. zu bestimmen, ob die Daten für den PSG selber bestimmt sind, oder ob er sie nur weiterleiten soll, benötigt man noch einige Kontrollsignale. Diese werden im Falle des Schneider-Computers in den oberen beiden Bits von Port C an den Sound Chip übergeben. Auf die Programmierung dieses Gerätes wird im Kapitel 1.5 noch näher eingegangen werden.

1.4.4.3 Der 8255 als Cassetten-Interface

Eine weitere wesentliche Aufgabe der 8255-PIO besteht in der Verbindung des Computers mit der Cassetteneinheit. Diese erhält alles in allem sechs Signale, von denen jedoch nur drei durch die PIO gestellt werden.

1. **Die Stromversorgung, +5V und GND:** Dies sind keine eigentlichen Signale. Die Zentraleinheit gibt sie jedoch an den Recorder aus, weshalb sie hier aufgeführt sind.
2. **SOUND:** Dieser Ausgang zum Recorder ist dem gleichnamigen Pin des Expansion Ports equivalent. Er ist, kurz gesagt, eine Zusammenschaltung der drei Tonkanäle des PSG (jeweils über 10kOhm) und ermöglicht die Aufzeichnung der Tonausgabe des PSG auf Cassette (anstelle der digitalen Datensignale).
3. **WR DATA:** Dieses Signal dient der Aufzeichnung digitaler (Daten-) Impulse auf Band. Es wird durch b5 von Port C des 8255 geliefert.
4. **RD DATA:** Dieses Signal entspricht den gelesenen, digitalisierten Signalen von der Bandeinheit. Es geht direkt an b7 von Port B des 8255. Dort kann es leicht getestet werden, indem der Port geladen und dann das Vorzeichen getestet wird.
5. **-MOTOR:** Obschon dieses Signal "low"-aktiv ist, wird es durch eine 1 in b4, Port C des 8255 aktiviert (Die Ausgabe wird dann noch einmal invertiert). Wie schon unschwer aus dem Namen zu

erkennen ist, dient dieses Signal dazu, den Cassetten-Motor einzuschalten.

1.4.4.4 Sonstige 8255-Bits

Neben den bisher aufgeführten gibt es noch einige der 24 I/O-Bits, auf die wir bisher noch nicht eingegangen sind.

VSYNC: Dieser Eingang (b0, Port B) wird vom CRTC bei einer vertikalen Synchronisation auf "high" gelegt. Das passiert immer genau dann, wenn der Bildaufbau beendet ist und der Elektronenstrahl von der rechten unteren Ecke in die linke obere (der Ausgangsposition) zurückfährt. Dieses Signal kann z.B. dazu benutzt werden, um den Programmablauf mit dem Bildaufbau zu synchronisieren. (In der Verarbeitung von Events wird dieses Signal zum Bestimmen eines Frame Fly's - so der englische Ausdruck dafür - benutzt, um festzustellen, wann die Frame Fly Chain bearbeitet werden muß.)

LK1-LK3: Diese drei Eingänge (b1-b3, Port B) werden durch feste Brükken jeweils entweder auf "high" oder auf "low" gelegt, um den jeweiligen Distributor des CPC-Computers in den verschiedenen Ländern zu ermitteln und in der Einschaltmeldung entsprechend zu berücksichtigen (so z.B. Amstrad in Great Britain, Schneider in der BRD).

LK4: Dieser Eingang (b4, Port B) ist ebenso wie die drei eben genannten an eine feste Brücke angeschlossen. Sein Zustand entscheidet jedoch nicht über den Firmennamen oder ähnliches, das bei der Einschaltmeldung ausgegeben werden muß. Vielmehr wird danach der CRTC auf entweder 50Hz (PAL-Norm) oder 60Hz (SECAM-Norm) Bildwiederholfrequenz initialisiert. In der Bundesrepublik wird allgemein das PAL-System verwendet, in Frankreich z.B. gilt die SECAM-Norm.

-EXP: Dieser Eingang (b5, Port B) wird durch einen Pin des Expansion Ports an den 8255 gegeben, und kann genutzt werden, um der CPU externe Erweiterungen anzuzeigen.

BUSY: Auch dieses Bit ist ein Eingang (b6, Port B). Es wird im Rahmen der Kommunikation mit einem Drucker benötigt und teilt dem Computer gegebenenfalls mit, ob der Drucker bereit ist, neue Zeichen zu empfangen, oder ob er noch mit dem Ausdruck anderer beschäftigt ist. Der Drucker ist empfangbereit, wenn das Signal "low" ist.

1.4.5 Pinbelegung der 8255-PIO

PA7-PA0: Dies sind die Herausführungen der Bits des Ports A. Sie können sowohl als Eingänge als auch als Ausgänge programmiert werden.

PB7-PB0: Wie oben, für Port B.

PC7-PC0: Wie oben, für Port C.

DA7-DA0: Dies ist die Verbindung zwischen dem PIO und der CPU. Im Schneider- Computer sind diese Pins an den Datenbus angeschlossen.

A1,A0: Diese beiden Pins dienen der Auswahl der internen Register in der PIO. Eine Auswahltabelle finden Sie im Abschnitt 1.4.3.

-CS (Chip Select): Dieses "low"-aktive Signal wird benutzt, um die PIO zu aktivieren. Ist es "high", so ist der PIO-Datenbus hochohmig.

-RD (ReaD): Dieses "low"-aktive Signal zeigt einen Lesezugriff auf ein PIO-Register an.

-WR (WRite): Dieses "low"-aktive Signal zeigt einen Schreibzugriff auf ein PIO-Register an.

RESET: Dieses Signal setzt die PIO in einen definierten Ausgangszustand zurück.

Vcc: Dies ist der Eingang für die Versorgungsspannung von normalerweise +5V gegenüber GND.

GND: Dies ist der Eingang für die Grundspannung, die per Definition einen Wert von 0 V hat.

1.5 Der programmierbare Sound-Generator AY-3-8912

1.5.1 Allgemeines

Der 8912 PSG (*P*rogrammable *S*ound *G*enerator, Anhang A5.1) ist der Chip, der im CPC für die Erzeugung der Klänge verantwortlich zeichnet. Er ist über die PIO an die Zentraleinheit angeschlossen und muß dementsprechend auch über den 8255 programmiert werden. Der PSG verfügt über drei Kanäle, die nahezu völlig unabhängig voneinander programmiert werden können. Für jeden Kanal kann einzeln die Frequenz und die Lautstärke eingestellt werden. Lediglich den Hüllkurven- und den Rauschgenerator müssen alle drei Kanäle zusammen benutzen, da es diese beiden Funktionseinheiten jeweils nur einmal auf dem Chip gibt.

Ein für einen Sound-Chip recht ungewöhnliches Feature ist der eingebaute I/O-Port. Dies läßt sich nur mit der Geschichte des 8912 erklären: Er wurde als als Sound-Chip für Telespiele entwickelt. Um in solchen Geräten die Bauteilzahl möglichst niedrig zu halten, wurden dem 8912 noch zwei recht einfache Ports für die Erledigung einfachster I/O-Prozesse mitgegeben. Beim 8912 ist jedoch nur ein Port (Port A) herausgeführt, um den Chip in ein möglichst kleines Gehäuse einbauen zu können. In diesem Fall hat der PSG 28 Pins. Bei seinem großen Bruder, dem AY-3-8910, sind dann beide Ports als Pins nach außen geführt, der Chip jedoch ist der gleiche.

1.5.2 Der Zugriff auf die PSG-Register

Der PSG verfügt über 16 interne Datenregister, auf die der Anwender zugreifen kann. Eines davon, das Datenregister für Port B, kann man jedoch nicht benutzen. Mittels dieser Register ist es möglich, auf die Generierung des Klanges Einfluß zu nehmen. Es handelt sich bei allen um 8-Bit Register. Der PSG verfügt zur Aufnahme von Daten über einen Datenbus, der im CPC mit dem Port B der 8255 PIO verbunden ist.

Es stellt sich nun jedoch die Frage, wie man als Programmierer das Register auswählt, in das man einen bestimmten Wert schreiben möchte. Außer den acht Datenleitungen stehen dem Programmierer nur noch zwei Verbindungen zum PSG zur Verfügung, auf die er Einfluß nehmen kann: 1. BDIR (Bus DIRection), an b7 von Port C des 8255 angeschlossen und 2. BC1 (Bus Control 1), verbunden mit b6 von Port C des 8255. Hinzu kommt im allgemeinen noch der Steueranschluß BC2. Im CPC ist dieser Pin jedoch permanent auf +5V gelegt, so daß er keine Funktionalität mehr besitzt. Die

beiden verbleibenden Bits können insgesamt vier verschiedene Zustandskombinationen aufweisen, die folgende Funktionen haben:

BDIR	BC1	Funktion
0	0	inactive
0	1	read
1	0	write
1	1	latch address

INACTIVE: Der Datenbus des PSG wird hochohmig, und der PSG akzeptiert auch keine Daten mehr.

READ: Die CPU kann jetzt (über den 8255) aus dem ausgewählten Register des PSG lesen.

WRITE: Die CPU kann in das jeweils ausgewählte Register des PSG schreiben.

LATCH ADDRESS: Wenn die CPU jetzt über den Port B Werte an den PSG ausgibt, so werden diese nicht mehr als Daten in ein Datenregister geschrieben, sondern als neue Registernummer interpretiert. Die CPU kann auf diesem Wege das Register des PSG auswählen, mit dem sie gerade arbeiten möchte. Ähnliche Verfahren zur Registerauswahl kann man auch in anderen Bausteinen des CPC finden, z.B. im CRTC und auch im Gate Array. Ist eine Registernummer einmal übergeben, dann bleibt das Register solange ausgewählt, bis ein neues Register bestimmt wird. Das relativ unkomfortable Arbeiten mit dem PSG wird dadurch doch um einiges erleichtert, da aufeinander folgende Zugriffe auf dasselbe Register mit dieser Technik recht einfach vonstatten gehen.

1.5.3 Die Bedeutung der PSG-Register

Nachdem wir nun wissen, was wir tun müssen, um auf die einzelnen Register zugreifen zu können, wollen wir hier endlich auf die Funktion sämtlicher PSG-Register eingehen.

REG #0/1: Dieses Registerpaar gibt die Periodendauer des Tons über Kanal A an. Die absolute Periodendauer ergibt sich aus der Zahl, die in diese Register geschrieben wird, multipliziert mit 1/62500 s (im Gegensatz zu den Angaben, die das Handbuch dazu macht). Es werden jedoch nicht alle 16 Bits genutzt, sondern nur 12: alle 8 Bits des Registers 0 (als Lo-Byte) und die vier unteren Bits des Registers 1 (als High-Nibble).

REG #2/3: gleiche Funktion wie Register 0/1, jedoch für Kanal B.

REG #4/5: gleiche Funktion wie Register 0/1, jedoch für Kanal C.

REG #6: Die unteren fünf Bit bestimmen die Periodendauer (durchschnittlich) des Rausch-Generators: je größer der Wert, desto länger die mittlere Periodendauer des Rauschens.

REG #7: Dies ist das Kontrollregister des AY 3-8912. Den einzelnen Bits sind hier jeweils verschiedene Funktionen zugeordnet:

b7:	Port B Datenrichtung	1=output	0=input
b6:	Port A Datenrichtung	1=outpu	0=input
b5:	Rauschen Kanal C	1=aus	0=an
b4:	Rauschen Kanal B	1=aus	0=an
b3:	Rauschen Kanal A	1=aus	0=an
b2:	Kanal C, Ton	1=aus	0=an
b1:	Kanal B, Ton	1=aus	0=an
b0:	Kanal A, Ton	1=aus	0=an

REG #8: Die unteren vier Bits bestimmen die Lautstärke, mit der der Ton von Kanal A ausgegeben wird, d.h. die Amplitude des Signals am Ausgang ANALOG A. Die Amplitude nimmt logarithmisch zu, so daß von menschlichen Gehör ein linearer Lautstärke-Anstieg wahrgenommen wird. Ein Sonderfall in diesem Register tritt dann ein, wenn b4 = 1 ist: Die Lautstärke des Tons wird dann nicht mehr durch die unteren vier Bits bestimmt, sondern durch den Hüllkurvengenerator, der im PSG eingebaut ist.

REG #9: Funktion wie Register 8, jedoch für Kanal B.

REG #10: Funktion wie Register 8, jedoch für Kanal C.

REG #11/12: Dieses Registerpaar bestimmt mit allen 16 Bits die Periodendauer der vom Hüllkurvengenerator erzeugten Hüllkurve. Register 11 ist dabei das Lo-Byte, Register 12 das Hi-Byte.

REG #13: Die unteren vier Bits dieses Registers bestimmen die Form der erzeugten Hüllkurve. Die Hüllkurven sind dabei nicht schlicht durchnummeriert, sondern die einzelnen Bits haben durchaus eine feste Funktion, woraus sich auch eine Systematik bei den Hüllkurven ergibt. Wir verzichten hier darauf, diese darzustellen. Stattdessen haben wir alle möglichen Hüllkurven mit den entsprechenden Bit-Kombinationen im Anhang A5.3 aufgeführt.

REG #14: Peripheres Datenregister des Port A. Je nach Zustand des entsprechenden Kontroll-Bits liegt unter dieser Registernummer entweder das Eingabe- oder das Ausgaberegister des Ports.

REG #15: Peripheres Datenregister des Port B. Je nach Zustand des entsprechenden Kontroll-Bits liegt unter dieser Registernummer entweder das

Eingabe- oder das Ausgaberegister des Ports. (Beim AY 3-8912 im Schneider-Computer ist dieser Port nicht nach außen geführt, was ihn weitgehend unbenutzbar macht.)

1.5.4 Die Programmierung eines Tons mit dem PSG

Nachdem uns nun die Bedeutung der Register bekannt ist und wir wissen, wie wir auf sie zugreifen können, sollten wir hier noch einmal kurz die Programmierung eines Tones darstellen. Wir benutzen auf Assemblerebene dazu die Routine MC SOUND REGISTER, die wir über ihrem RAM-Vektor bei $BD34 anspringen (im O.S.-ROM liegt sie bei $0826). Diese Routine erleichtert uns den Zugriff auf die PSG-Register, der - wie wir ja wissen - sonst nur über mehrfaches Umschalten der beiden Steuersignale von Port C aus möglich ist. Dieser Routine muß im Akku die Registernummer, in C das Byte, das man in das Register zu schreiben wünscht, übergeben werden.

Wir wollen jetzt einen Ton von 1000 Hz, ohne Rauschen und mit der Hüllkurve 1-1-1-0 ausgeben. Dazu müssen wir zunächst den Wert ermitteln, den wir als Periodendauer setzen, um die Frequenz von 1000 Hz, d.h. die Periodendauer von 1/1000 s zu erreichen. Dieser Wert ergibt sich aus der Division der erwünschten Periodendauer durch die Periodendauer der Grundschwingung (1/62500 s), in unserem Fall also (1/1000s)/(1/62500s) = 62,5. Als Periodendauer ergibt sich somit entweder der Wert 62 oder 63. Wir wählen willkürlich eine Periodendauer von 63 Einheiten.

Die Werte, die nun dem PSG zu übergeben sind, ergeben sich aus dem vorher gesagten: Nehmen wir an, wir möchten den Ton über Kanal A ausgeben. Der Einfachheit halber gehen wir gleich davon aus, daß die anderen Kanäle zur Zeit nichts sinnvolles mehr tun. Um nun Kanal A anzuschalten, schreiben wir das Byte $3E (binär %00111110) in das Kontroll-Register (Register 7) des PSG. Die errechnete Periode (63, $3F) bringen wir in den Registern 0 und 1 unter - in Register 1 das Hi-Byte (in unserem Fall natürlich 0), in Register 0 das "Lo"-Byte (also hier $3F). Da wir die Lautstärke vom Hüllkurven-Generator regeln lassen wollen, setzen wir als Kennzeichen dafür b4 des Lautstärke-Registers von Kanal A (Register 8). Zu guter Letzt legen wir noch die Hüllkurve fest. Ihre Form hatten wir bereits ausgewählt mit der Bitkombination 1-1-1-0. In Register 13 schreiben wir daher den Wert $0E (%00001110). In das Registerpaar 11/12 müssen wir die Periodendauer der Hüllkurve schreiben, so z.B. $C000. In Register 12 muß dann der Wert $C0, in Register 11 eine Null gebracht werden.

Das entsprechende Assembler-Programm ist sehr einfach zu erstellen. Es erscheint uns nicht sinnvoll, ein Listing an dieser Stelle einzufügen (zumal

dieses Beispiel wirklich sehr simpel und speziell ist). Die Erzeugung von Klängen kann sehr verschieden verwaltet werden: die unterste Stufe wäre ein Programm, das tatsächlich nur einen festen Ton ausgeben kann. Der Sound Manager des CPC hingegen läßt die Handhabung des PSG auf einer extrem anwenderfreundlichen Ebene zu. Dies gilt natürlich auch für den Anwender auf Assembler-Niveau. Im Sound Manager sind alle wesentlichen Features realisiert, die man bei der Kontrolle des AY 3-8912 sinnvoll einsetzen kann. Das Listing dieses Betriebssystem-Segments ist deshalb auch als Lektüre für den engagierten Sound-Programmierer zu empfehlen - es steckt voller Anwendungsbeispiele.

1.5.5 Pinbelegung des AY 3-8912

DA7-DA0: Dies sind bidirektionale Datenverbindungen zum Computer. Die CPU schreibt und liest über diese Leitungen Werte in und aus dem PSG.

BC1, BC2 (Bus Control): Diese beiden Pins dienen der Kontrolle des Datenverkehrs zwischen CPU und PSG. Da jedoch einige Funktionen (in Verbindung mit dem BDIR-Pin) doppelt vorkommen, hat man - um so die Redundanz zu senken - BC2 ständig auf "high" (+5V) gelegt.

BDIR (Bus DIRection): Dieser Pin kontrolliert ganz allgemein die Richtung des Datenflusses über den PSG-Datenbus. In Verbindung mit den BC-Pins hat er ganz bestimmte Funktionen, die ausführlich in Abschnitt 1.5.2 erläutert sind.

A8: Dies ist das Signal, das den PSG erst aktiviert, d.h. auswählt (es ist daher ein Chip-Auswahl-Signal). Es ist"high"-aktiv und im CPC immer auf +5V gelegt, so daß der PSG dauernd aktiviert ist.

-RESET: Dieses "low"-aktive Signal initialisiert den PSG, d.h. es unterbricht seine Aktivitäten und setzt seine Register zurück. Es ist an das normale Reset-Signal im CPC angeschlossen.

CLOCK: An diesem Pin liegt der Master-Takt an, d.h. der Takt, aus dem durch Frequenzteilung der Grundtakt für die Tonerzeugung gewonnen wird. Im Schneider-Computer wird dieser Pin konstant mit einem Takt vom 1 MHz versorgt. Durch einen 4-Bit-Frequenzteiler wird diese Frequenz durch 16 geteilt und als Grundschwingung an die tonerzeugende Einheit im PSG weitergegeben. Diese Grundschwingung hat damit eine Frequenz von 1 000 000/16 = 62 500 Hz, weshalb die Periodendauer auch in Einheiten von 1/62500 s angegeben werden muß.

ANALOG CHANNEL A,B,C: An diesen drei Ausgängen des PSG liegen schließlich die Tonschwingungen der drei verschiedenen Kanäle, repräsen-

tiert durch Spannungen. Die maximale Spannung, die hier anliegen kann, ist die Versorgungsspannung (Vcc).

Vcc: An diesem Pin liegt die Versorgungsspannung, normalerweise +5V.

Vss (oder auch GND): Hier liegt die Grundspannung, relativ 0 V.

IOA7-IOA0: Dies sind die Portbits des Ports A, die sowohl als Eingänge, wie auch als Ausgänge programmiert sein können. (Beim Schwesterchip 8910 gibt es die entsprechenden Pins auch noch für den Port B.)

TEST 2: Dieser Pin dient dem Testen des Chips bei der Herstellung. Vom User kann er nicht sinnvoll verwendet werden und ist im Schneider CPC daher auch unbeschaltet geblieben.



2 Grundlegende Strukturen

An dieser Stelle möchten wir Ihnen einige der grundlegenden Strukturen, die im Schneider Computer ihre Anwendung finden, ein wenig näherbringen. Zwar sind sie zur Nutzung der Routinen der CPC-ROMs nicht nötig, will man jedoch das Betriebssystem oder das Basic des CPC richtig verstehen, so ist es unerläßlich, auch die ihnen zugrunde liegenden Strukturen zu begreifen. Auch können die Datenstrukturen und Programmiertechniken, die wir hier vorstellen wollen, eine Anregung für das Schreiben eigener Programme sein, da nicht nur der Algorithmus über den Wert eines Programmes entscheidet, sondern beispielsweise auch die Datenstrukturen, die es verwendet. Wir können natürlich im Rahmen dieses Buches ein Thema wie Datenstrukturen nicht umfassend behandeln - ganze Regale von Büchern sind darüber geschrieben worden. Dies kann nur ein Anstoß sein, das faszinierende Gebiet näher kennenzulernen.

2.1 Datenspeicherung

Wir unterscheiden in diesem Kapitel streng die Datenstrukturen von der Datenspeicherung (Memory Allocation). Das ist umso wichtiger, als auch erfahrene Programmierer nur allzu gerne diese völlig verschiedenen Dinge im einen Topf werfen. Dabei handelt es sich bei der Datenspeicherung um die Anordnung der Daten im Speicher bzw. um deren gegenseitige Verkettung durch Adressen, Nummern und ähnlichem. Datenstrukturen (Data Structures) hingegen beziehen sich ausschließlich auf die *logische* Struktur der Daten, d.h. auf deren logische Beziehung untereinander, die durchaus von ihrer speicherplatzmäßigen Beziehung abweichen kann. Datenstrukturen sind also gewissermaßen die Summe aller Ideen, die hinter der Anordnung der Daten eines Programmes stehen. Die Datenspeicherung ist "nur" die Realisierung dieser Ideen im Rahmen eines Computers. Diese beiden Qualitäten werden deshalb so oft durcheinander gewürfelt, weil es oft günstig ist, die Beziehungen anzugleichen, die auf den beiden Ebenen (Idee-Speicher) bestehen.

2.1.1 Records

Der Begriff des Records ist ein zentraler Begriff, sowohl für die Datenspeicherung, als auch für Datenstrukturen. Innerhalb einer Struktur ist ein Record eine Speichereinheit, in gewissem Sinne also ein Speicheratom. Dieser Vergleich hinkt jedoch, da ein Record selber beliebig komplex sein kann, selber also aus riesigen Datenstrukturen bestehen kann. Innerhalb

seiner Struktur - auf der *Ebene* dieser Struktur also - ist der Record nach außen hin unstrukturiert, wie bereits oben gesagt, eine Einheit.

So kann ein Record z.B. ein String sein, der selber über Unterstrukturen (der Länge und einer variablen Anzahl von Zeichen) verfügt. Auch eine 2-Byte-Integerzahl innerhalb eines Feldes beispielsweise ist ein Record, obschon er (semantisch) nicht weiter untergliedert werden kann. In anderen Sprachen gibt es z.B. Daten, die nur ein Bit benötigen (Flags); auch solch simple Daten sind logisch Records. Records können aber auch hochkomplexe Gebilde sein, z.B. ein Feld mit 200 mal 200 Integerzahlen. Ein solcher Record würde 80000 Bytes umfassen, wäre logisch aber nichts weiter als das Flag oder die eine Integerzahl.

Die unbegrenzte Vielfalt von Records ordnet man nun zunächst einmal in zwei globale Kategorien. Wir werden sehen, daß diese Einteilung noch weitreichende Folgen hat. Es gibt solche mit einer festen Länge (im Speicher), die "fixed length records" (kurz FLR), und solche mit einer variablen Länge (im Speicher), die "variable length records" (kurz VLR). An dieser Stelle erkennt man deutlich den Unterschied zwischen Datenspeicherung und Datenstrukturen: Für die Struktur der Daten ist diese Eigenschaft der Records vollständig unerheblich, während es zu den Aufgaben der Datenspeicherung gehört, sich mit derlei Problemen zu befassen.

Betrachten wir an dieser Stelle einmal die Eigenschaften von VLRs und FLRs. FLRs haben den Vorteil, daß der Zugriff auf sie wesentlich unproblematischer verläuft. Sucht man z.B. in einer Tabelle aus FLRs das n-te Element, so kann man die Adresse des Records leicht mit $a = b + l*(n - 1)$ berechnen, wobei a die Record-Adresse, l die Länge der Records und b die Basisadresse der Tabelle ist. Die Suche nach einem VLR würde sich unangenehmer gestalten, da man alle Elemente in der Tabelle durchgehen müßte, um das gesuchte Element schließlich zu finden.

Nun ist durch die Art der zu speichernden Daten in gewisser Weise schon vorgegeben, welche Art von Records zur Speicherung zu verwenden sind. 5-Byte-Fließkommazahlen, 2-Byte-Integerwerte werden natürlich in FLRs abgelegt, wohingegen sich für Strings z.B. aufgrund ihrer unbestimmten Länge VLRs zur Speicherung anbieten würden.

Da jedoch der Zugriff auf FLRs so viel schneller ist, als der auf VLRs, geht man in vielen Sprachen andere Wege, indem man VLRs möglichst vermeidet. Im UCSD-Pascal-System z.B. werden Strings auf eine feste Länge gesetzt (der Default ist 80 Zeichen) und grundsätzlich in Speicherbereiche mit konstanter Länge gespeichert (der Default ist 81 Bytes: 80 Zeichen + 1 Byte Count, das die Länge des eigentlichen Strings

bestimmt). Diese Art der Speicherung ist zwar wesentlich schneller und unproblematischer als die Strings als VLRs abzulegen, man erkennt jedoch, daß sie auch sehr speicherplatzintensiv ist.

Auch im Basic "verwandelt" man die VLRs in FLRs, indem man in die Variableneinträge der Basic-Variablen nicht den String speichert, sondern einen Zeiger auf den String (der wegen seiner festen Länge von 2 Byte ja ein FLR ist). Die eigentlichen Strings werden in gepackter Form (d.h. String an String ohne ungenutzte Leerstellen) am Ende des Speichers abgelegt. Diese Form der Speicherung verschwendet nicht so viel Speicherplatz, wie die, die UCSD-Pascal für Strings verwendet, und ist nur unwesentlich langsamer. Basic hat dort also einen Kompromiß gefunden.

2.1.2 Arrays (Felder)

Arrays sind Ihnen vermutlich schon aus der Arbeit mit Basic ein Begriff. Neben den einfachen, unstrukturierten Variablen sind Arrays die einzige Art der Datenspeicherung, die von Basic unterstützt wird.

Arrays dienen hauptsächlich zur Speicherung von FLRs. Das herausragende Merkmal von Arrays ist die Tatsache, daß die Records konsekutiv, d.h. hintereinander angeordnet sind. Will man also in einem Programm mehrere Records in einem Array ablegen, so muß der Programmteil, dem ein Record übergeben wird, dieses in den Bereich speichern, der für die Arrays vorgesehen ist. Dieses Kopieren eines Datenblocks kann bei größeren Blöcken schon einen beachtlichen Zeitaufwand bedeuten. Auch sind die Daten an eine bestimmte Position gebunden - nämlich an den Speicherbereich, in dem das Array realisiert wird. Vielfach ist es aber sinnvoll, Records positionsunabhängig (position independent) im Speicher zu organisieren. Aber dazu kommen wir später.

Arrays haben aber noch andere Nachteile. So stellt z.B. das Einfügen bzw. Löschen eines Elementes in einem Array ein großes Problem dar, da die darüber liegenden Datenblöcke allesamt verschoben werden müssen. Dies gilt besonders für Arrays mit VLRs. In diesem Fall haben die Einträge jeweils eine unbestimmte Länge, d.h. aus der Nummer des Elements kann dessen Adresse nicht einfach berechnet werden. Die Trennung der einzelnen VLRs kann auf ganz verschiedene Arten erfolgen. So z.B. dadurch, daß die Länge des Records mit abgespeichert wird. Um auf ein Element zugreifen zu können, muß das Programm also alle davor liegenden Elemente auf ihre Länge untersuchen und diese Länge zur laufenden Adresse addieren. Dies ergibt dann die Adresse des folgenden Records und so weiter. Records können aber auch dadurch separiert werden, daß man an den Anfang oder an das Ende eines Datenblockes jeweils eine Markierung setzt

(d.h. ein Byte oder eine Folge von Bytes, die in den Daten nicht vorkommen kann). Vielfach werden Strings in dieser Form abgespeichert. So werden UNIX-Texte mit einem LF-Zeichen als Zeilenterminator abgespeichert. CP/M benutzt ein CR, die Sprache C eine Null.

Arrays haben natürlich auch Vorteile. Bei FLRs liegen diese klar auf der Hand: Der Zugriff auf ein Element ist mit sehr wenig Aufwand nach der bereits aufgeführten Formel für die Adresse a = b + (n - 1) * l möglich. Wenn die Indizes - so der Name für die Nummer eines Elements - bei 0 anfangen, verkürzt sich die Formel gar auf a = b + n * l. Desweiteren kann es sinnvoll sein, bestimmte Daten eines Systems innerhalb eines Speicherbereichs möglichst ohne Leerstellen abzuspeichern. So ist dies z.B. im Stringbereich des Basic der Fall, in dem die Strings ja bekanntlich in Form eines Arrays (d.h. direkt hintereinander) abgelegt sind. Es ist kaum möglich, sich eine andere Struktur für diesen Anwendungsfall vorzustellen.

Es gibt im Schneider die unterschiedlichsten Formen von Arrays. So sind z.B. die Adressentabellen des Basic für Funktionen und Befehle FLR-Arrays, wohingegen die entsprechende ASCII-Tabelle für die Keywords ein VLR-Array mit einer recht komplexen logischen Struktur darstellt.

2.1.3 Linked Lists (verkettete Listen)

Eine Linked List ist eine Art der Datenspeicherung, bei der die Speicherposition der Records keine Rolle mehr spielt. Dies geschieht dadurch, daß zusätzlich zu dem eigentlichen Record in jedem Eintrag ein Zeiger auf den nächsten Eintrag enthalten ist. Dieser Zeiger kann auf ganz unterschiedliche Arten zu dem folgenden Record führen. Häufig wird einfach die Adresse des Records dort abgelegt, es können aber auch irgendwelche Nummern oder Indizes sein, mit denen sich der folgende Eintrag dann schließlich irgendwie lokalisieren läßt. Wenn wir also im Folgenden von einem "Zeiger" (Pointer) reden, so meinen wir stets den allgemeinen Verweis auf ein Element, also einen beliebigen Weg, der es möglich macht, den nächsten Record eindeutig im Speicher aufzufinden. Die Adresse ist stets der Spezialfall eines solchen Zeigers. Zur theoretischen Betrachtung sei noch gesagt, daß die Records einer Linked List alle in einem - durch die Art des Zeigers festgelegten - Speicherbereich liegen müssen. Dieser Bereich *kann* den gesamten Adreßraum des jeweiligen Systems umfassen, es ist jedoch ebensogut denkbar, daß nur ein Teil des gesamten Speichers einer Linked List zur Verfügung steht.

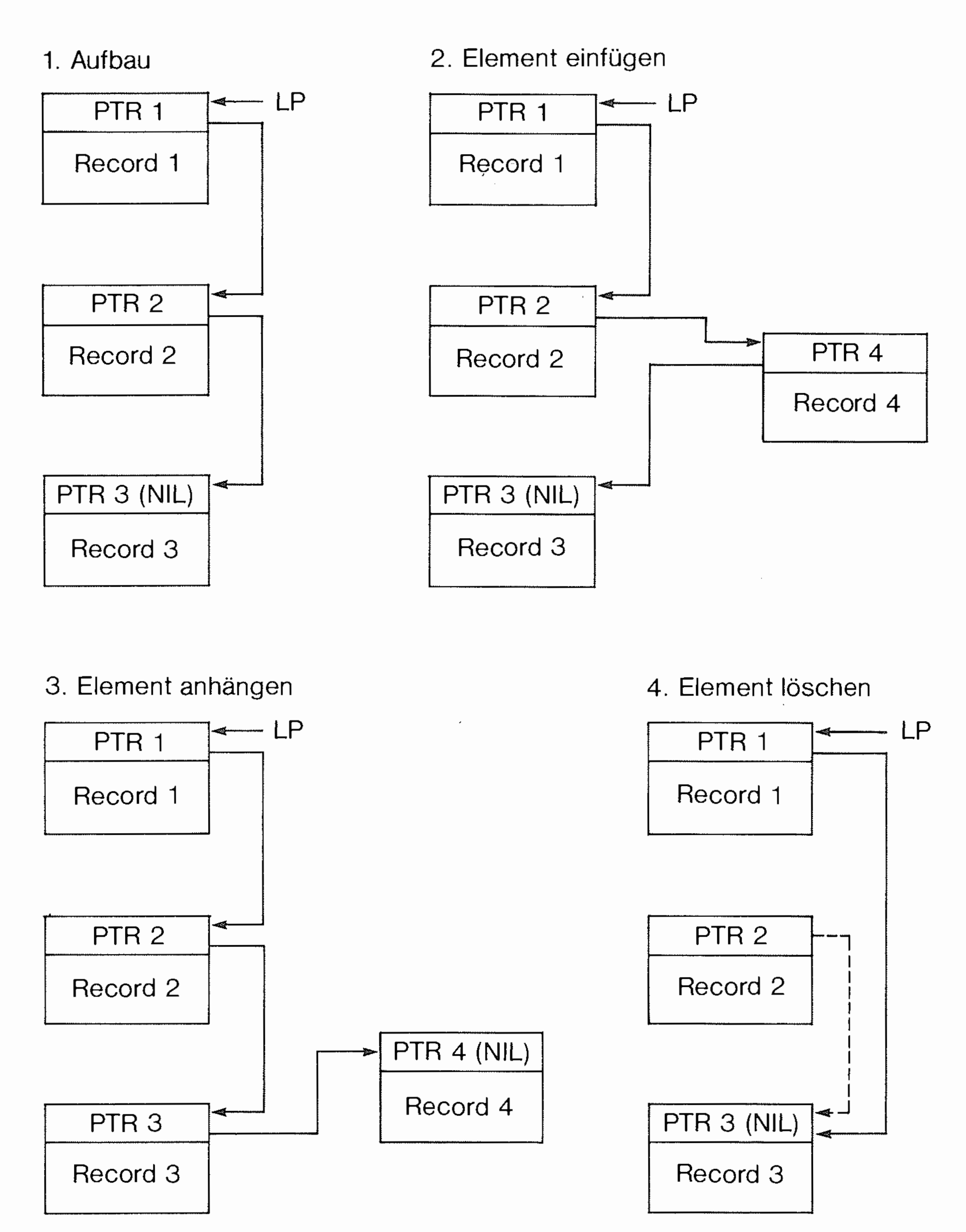


Abbildung 2.1: *LINKED LIST - Verkettete Liste (VL)*

In Abbildung 2.1 sehen Sie in der Darstellung 1 das Modell einer Linked List. Der einzige Platz, der zur Definition einer verketteten Liste notwendig ist, ist der Ort des "List Pointers", d.h. die Adresse (oder das Register), in der der Zeiger auf das erste Element der Liste steht. Alles andere "ergibt" sich förmlich aus der Liste selber: die Elemente der Liste zeigen alle auf das jeweils folgende Element - bis auf den letzten Eintrag. Sein Zeiger muß derart beschaffen sein, daß aus ihm geschlossen werden kann, daß das Programm am Ende der Liste angelangt ist. Das wird in vielen Fällen dadurch erreicht, daß dieser Zeiger einen besonderen Wert erhält, den ein normaler Zeiger (aus irgendwelchen Gegebenheiten des jeweiligen Systems) nicht annehmen kann. Diesen Wert nennt man dann NIL (Not In List). Das NIL ist eine logische Struktur, die oftmals durch eine Null realisiert wird. Bei den meisten Systemen kann eine Null als Zeiger nicht auftreten, da der untere Bereich meist für Systemvariablen gebraucht wird. Viele Programmierer verwechseln diese unterschiedlichen Begriffe NIL und Null.

Wir hatten schon gesagt, daß die Linked List eine Art von Positionsunabhängigkeit (Position Independence) der Records ermöglicht. Dies ist oft ein Vorteil, kann aber auch störend sein. Man kann diese Art der Datenspeicherung aber auch noch unter anderen Aspekten betrachten. So ist es zunächst einmal recht unwichtig, ob sie mit FLRs oder mit VLRs verwirklicht wird - bei beiden ist der Zugriff auf einen Eintrag identisch. Dieser Zugriff nun ist zwar schneller, als der Zugriff auf VLR-Arrays mit Separatoren, jedoch bedeutend langsamer als auf FLR-Arrays, da auch hier alle Records vor dem gesuchten Record durchgegangen werden müssen.

Die Vorteile einer Linked List kommen zum Tragen, wenn es darum geht, Elemente einzufügen oder zu löschen. Wir haben diese Prozesse in den Darstellungen 2 bis 4 der Abbildung 2.1 einmal anschaulich dargestellt. So kann z.B. Record 4 zwischen Record 2 und 3 eingefügt werden (Darstellung 2), indem PTR2 nach PTR4 kopiert wird und PTR2 schließlich auf den Record 4 gesetzt wird. Durch die Veränderung von nur 2 Zeigern wurden so ganze Datenblöcke miteinander verbunden. Ganz ähnlich wird auch ein Record an eine Liste angehängt (Darstellung 3). Der letzte Zeiger wird auf das neue Element gesetzt und in dessen Zeiger wird ein NIL geschrieben. Auch das Löschen eines Elementes ist sehr einfach (Darstellung 4). Der Zeiger des zu löschenden Elements wird in den vorhergehenden Zeiger kopiert (PTR2->PTR1). Das ist bereits der gesamte Vorgang, der nötig ist, um Record 2 aus der Linked List auszuhängen. Der Zeiger von Record 2 zeigt dann immer noch auf Record 3, was jedoch unerheblich für die Kette ist.

Die Linked List, die wir hier beschrieben haben, ist eine "lineare" Linked List, d.h. daß ein Eintrag jeweils immer nur auf den nächsten Eintrag zeigt und daß die Liste einen Anfang und ein Ende hat. Daneben gibt es noch sehr viele Variationen dieser Datenorganisation, z.B. die "zirkuläre" (circular) Linked List, in der der letzte Zeiger kein NIL enthält, sondern auf den ersten Eintrag der Liste zeigt. Dann ist auch die doppelt verkettete Liste (double linked list) eine recht häufig verwendete Organisationsform. Ein Eintrag enthält jeweils zwei Zeiger: einen auf das nächste, den zweiten auf das vorhergehende Element. Dies sind nur zwei der zahlreichen Erweiterungen des "Zeiger-Prinzips".

2.2 Datenstrukturen

Wie bereits erwähnt, beschreiben die Datenstrukturen die *Idee*, die hinter der Organisation der Daten sich verbirgt, d.h. die logische Beziehung zwischen den Daten. Im Gegensatz zur Datenspeicherung, die an sich eine statische Beziehung der Datenblöcke im Speicher definiert, gehört zur Betrachtung der Datenstrukturen immer die Betrachtung der Dynamik, d.h. der Art und Weise, in der die Daten verarbeitet werden. Man unterscheidet zwei grundsätzliche Strukturen, den LIFO und den FIFO.

2.2.1 Das LIFO-Prinzip (Stacks)

Die Bezeichnung LIFO kommt von "Last In - First Out", also zuletzt hinein, zuerst heraus. In der Umgangssprache werden sie auch durch den Begriff "Stack" (Stapel) beschrieben, obschon man durchaus die Synonymität dieser beiden Begriffe anzweifeln kann. Dennoch ist das Bild eines Stapels eine sehr gelungene Veranschaulichung der LIFO-Struktur. Nehmen wir zum Beispiel einen Bücherstapel. Das einzige Element dieses Stapels, auf das man direkt zugreifen kann, ist das oberste Buch, also der oberste Eintrag. Dies ist das Buch, das zuletzt auf den Stapel gelegt wurde, und es ist das, welches als erstes vom Stapel genommen werden wird, wenn wir uns jetzt daran machen, den Stack abzubauen. Auf diese Weise lassen sich genau zwei Operationen definieren, die man mit dem Stapel durchführen kann: ein neues Element oben auf den Stapel legen, was mit "push" bezeichnet wird, bzw. das oberste Element vom Stapel holen, was man "pop" oder "pull" nennt.

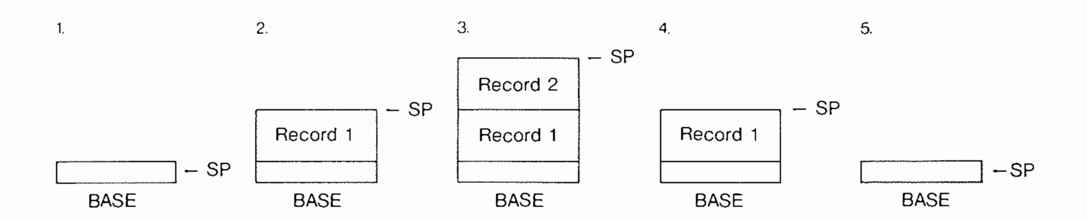


Abbildung 2.2: *STACK, feste Record-Länge (FLR)*

Wir haben diesen Vorgang in den Abbildungen 2.2 und 2.3 ins Bild gesetzt. In beiden Abbildungen werden auf einen zunächst leeren Stack zwei Records nacheinander gepusht und dann - in umgekehrter Reihenfolge - gepullt. Der Unterschied zwischen den beiden Abbildungen besteht darin,

daß in 2.2 die einzelnen Records hintereinander, in Form eines Arrays, abgespeichert sind (dies macht vor allem dann Sinn, wenn es sich bei den Records um FLRs handelt), während die Records in 2.3 in Form einer Linked List miteinander verbunden sind (was in dieser Anwendung vor allem bei VLRs vorkommen dürfte). Wir haben diese beiden Abbildungen deshalb aufgenommen, um noch einmal die (zumindest theoretische) Unabhängigkeit der Datenorganisation von den Datenstrukturen zu dokumentieren.

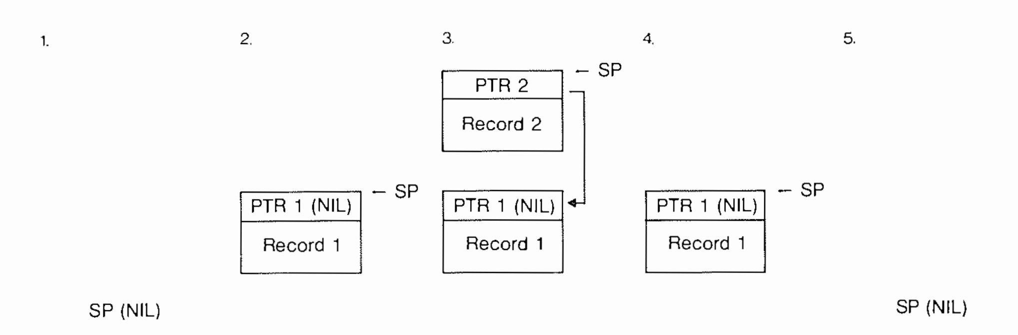


Abbildung 2.3: *STACK, variable Record-Länge (VLR)*

Es ist gut möglich, daß Sie mit der Struktur des Stacks schon vertraut waren. Auf Maschinenebene ist bereits ein sehr einfacher Stack durch die Hardware implementiert: der Prozessorstack, auf dem vor allem Rückkehradressen für CALLs und ähnliche Befehle, aber auch Daten abgelegt werden. Er wird durch den Stackpointer im Prozessor, das SP-Register realisiert. Der Hardware-Stack ist ein Stack des ersten Typs, da die Records eine feste Länge haben (2 Byte, wenn man einmal nur die RETurn-Adressen berücksichtigt) und da sie vor allem im Speicher aufeinander folgend abgespeichert werden (also in Form eines Arrays). Ein Beispiel für einen Stack der zweiten Art, der also in Form einer Linked List realisiert ist, finden Sie in Abschnitt 4.5.3 über die Auswertung einer User-Funktion in Basic.

2.2.2 Das FIFO-Prinzip (Queues)

Das Kürzel FIFO steht für "First In - First Out", also etwa zuerst hinein, zuerst heraus. Dieselbe Struktur wird bei Anwendungen, wo es um die Abwicklung von Anfragen externer Geräte bzw. um das Bedienen mehrerer Tasks in einem Multitasking-System geht, auch FCFS (First Come - First Served) genannt. Dies ist praktisch nichts weiter als die "Gegenstruktur" zum Stack. Der Eintrag, der als erster in eine solche Struktur übernommen

wurde, wird also auch als erster aus ihr wieder entfernt. Der umgangssprachliche Ausdruck dafür ist "Queue" (deutsch etwa "Schlange", im Sinne von: Reihe von Menschen). Tatsächlich ist der Supermarkt ein idealer Platz, um sich diese Datenstruktur zu veranschaulichen. Die Reihen, die sich an den Kassen bilden, werden nämlich (wenn man von einigen unrühmlichen Ausnahmen einmal absieht) ausschließlich nach dem FIFO-Prinzip bearbeitet: Wer als erster kommt, wird auch als erster bedient.

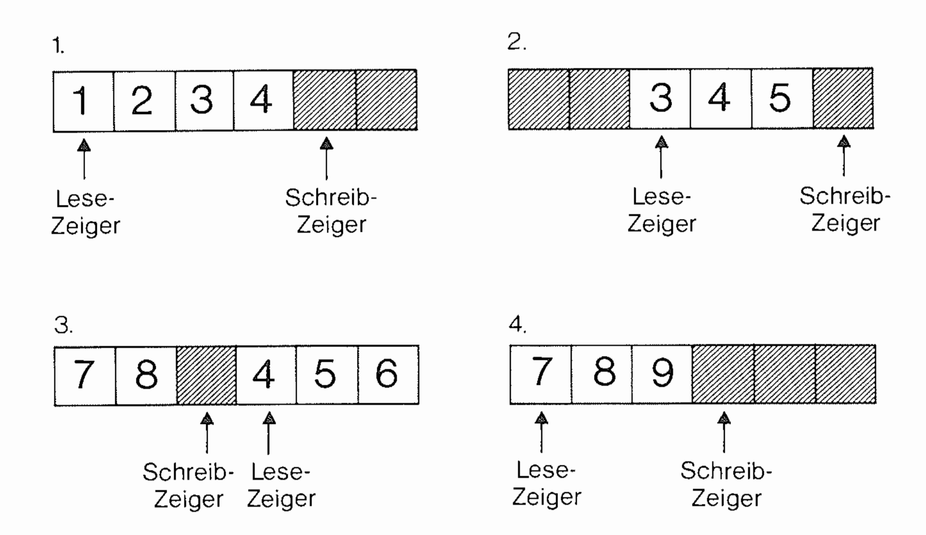


Abbildung 2.4: *Ringbuffer*

Beispiele für solche Queues gibt es viele im Schneider-Computer. Das wohl klassische Beispiel ist ein "Ringbuffer". Der Ringbuffer ist ein Speicherbereich, in den Records mit fester Länge geschrieben werden. Er hat also die Form eines Arrays. Da seine Größe fest ist und die Länge der Einträge ebenfalls definiert ist, ergibt sich daraus, daß ein Ringbuffer nur eine ganz bestimmte Anzahl von Einträgen aufnehmen kann. Wie die meisten Buffer, so verfügt auch der Ringbuffer über zwei Zeiger: einen, der auf das Element zeigt, das als nächstes zu lesen ist und einen, der auf die Stelle zeigt, in die das nächste Element geschrieben wird. Das Besondere an einem Ringbuffer ist nun aber, daß diese beiden Zeiger, wenn sie über das Ende des Bufferbereiches (im Speicher) hinausgehen, wieder auf den Bufferanfang gesetzt werden. Sie verfolgen sich demnach gegenseitig im Buffer und laufen in einer Art Kreis im Buffer herum.

Wir haben versucht, diesen Prozeß in Abbildung 2.4 zu verdeutlichen: Der Buffer dort kann maximal sechs Elemente aufnehmen. Zunächst enthält er vier Elemente. Dann wird eines (Nummer 5) hineingeschrieben und zwei werden herausgelesen (Nummer 1 und 2). Die schraffierten Flächen geben jeweils die unbenutzten Positionen im Buffer an. Dann werden als nächstes drei Records in den Buffer geschrieben (6,7 und 8) und ein Element (3) herausgelesen. Schließlich wird ein Element (9) geschrieben und drei (4,5,6) werden gelesen.

Der Ringbuffer tritt im CPC an zwei Stellen auf: im Keyboard Manager als Buffer zur Speicherung der Tastenkoordinaten und im Sound Manager zur Realisierung der Queue (d.h. der Warteschlange) der Töne für jeden Kanal. Im ersten Fall sind die Einträge zwei Byte lang und der Ringbuffer ist für 20 Einträge ausgelegt, im zweiten Fall sind die Einträge acht Byte lang, und es passen genau vier Records in jeden der drei Ringbuffer.

Eine Queue kann jedoch nicht nur über ein Array in Form eines Ringbuffers organisiert werden, auch die Linked List wird häufig dazu verwandt. Dies ist sehr einfach zu realisieren, indem man an das eine Ende der Liste nur Records anhängt und von dem anderen Ende nur Records entfernt. Im CPC kann man z.B. die Synchronous Pending Queue als eine solche Queue betrachten. Sie ist jedoch kein reiner FIFO: Die Events werden in ihr nicht nur an einem Ende eingehängt, sie werden vielmehr nach Prioritäten geordnet in die Queue gebracht.

Die Einführung einer Priorität ist nicht die einzige Modifikation, die man mit der FIFO-Struktur machen kann. So gibt es Queues, an denen bei beiden Enden sowohl Records eingehängt, wie auch ausgelesen werden können. Man nennt diese Struktur dann eine double-ended queue, oder auch dequeue (gesprochen "deck").

2.3 Programmstruktur und Programmiertechniken

Die Funktionsweise der einzelnen Routinen im ROM des CPC kann man trotz der umfassenden Dokumentation des ROM-Listings nicht nachvollziehen, wenn man die Programmstruktur nicht betrachtet. Im Folgenden werden einige Strukturen und Techniken erläutert, die nicht so einfach nachzuvollziehen sind bzw. die speziell im Schneider CPC eingesetzt werden.

2.3.1 Rekursion

Rekursion bedeutet wörtlich "Zurückgehen". Ein (Unter-)Programm ist dann rekursiv, wenn es an seinen Anfang zurückgeht. Eine anschaulichere Erklärung besagt, daß ein rekursives Unterprogramm sich selbst aufruft. Solch ein Aufruf bedeutet ja (unter anderem) ein Zurückgehen an den Anfang des Unterprogramms.

Zur Verdeutlichung ein Beispiel aus der Mathematik. Die Fakultät (n!) einer Zahl n ist das Produkt aller natürlichen Zahlen bis einschließlich zu der Zahl n. Demnach ist 4! (lies: vier Fakultät) also 1 * 2 * 3 * 4 = 24. Die Allgemeine Definition lautet: n! := 1*2*3* ... *n. Per Programm kann man die Fakultät mit einer einfachen Schleife berechnen:

```
100 REM FAKULTÄT N!=FAK(N)
110 FAK=1
120 FOR I=1 TO N
130 FAK=FAK*I
140 NEXT
150 RETURN
```

Die Fakultät läßt sich eleganter auch rekursiv definieren: n! := n * (n - 1)!. Die direkte Umsetzung in Basic ist hier zwar nicht so einfach möglich, in anderen Sprachen oder auch in Assembler läßt sich jedoch eine rekursive Lösung verwirklichen:

```
FAK         CP      2       ;Zahl <=1 ?
            LD      B,1     ;dann Resultat=1,
            RET     C       ;fertig
            PUSH    AF      ;Zahl retten
            DEC     A
            CALL    FAK     ;Fakultät von (Zahl-1)
            POP     AF      ;mit Zahl
            CALL    MULT    ;multiplizieren
            LD      B,A     ;Ergebnis nach B RET
```

Die Berechnung der Fakultät wird hier also auf sich selbst zurückgeführt: Zur Berechnung der Fakultät von n wird die Fakultät von n-1 berechnt und diese dann mit n multipliziert.

Es werden zwei Grundvoraussetzungen der Rekursion deutlich: Erstens muß eine rekursive Routine eine Abbruchbedingung enthalten, damit sie sich nicht unendlich oft selbst aufruft. Diese Abbruchbedingung ist in unserem Beispiel dann erfüllt, wenn die eingegebene Zahl kleiner oder gleich eins ist. Zweitens werden der rekursiven Routine ein oder mehrere Parameter übergeben, in unserem Beispiel eine Zahl in A. Parameter sind jedoch lokal zu der Routine, der sie übergeben werden. Lokal bedeutet in diesem Sinne, daß beim Aufruf der Routine ein eigener Platz für die Parameter reserviert wird, auf den die aufrufende Routine keinen Zugriff hat. Ein lokaler Parameter kann z.B. nicht in einer festen Speicherstelle gespeichert werden, auf die auch die aufrufende Routine zugreift. Allgemein zugreifbare Speicherstellen oder Variablen werden global genannt.

In unserem Beispiel wird die Lokalität des Parameters dadurch erreicht, daß der Akkumulator vor dem rekursiven Aufruf auf den Stack gerettet wird. Der Akkumulator kann dann von der aufgerufenen Routine für lokale Zwecke verwendet (und sein Inhalt zerstört) werden. Lokale Variablen und Parameter werden bei rekursiven (und auch bei nicht rekursiven) Routinen im allgemeinen auf einen Stack gerettet. Ein Stack ist aufgrund seiner LIFO-Struktur (siehe 2.2.1) für die Speicherung lokaler Variablen besonders geeignet, da die Parameter in der umgekehrten Reihenfolge wieder zurückgeholt werden müssen, in der sie auf den Stack gespeichert wurden.

Eine wichtige Vorraussetzung der Rekursion ist natürlich, daß die übergebenen Parameter sich mit der Rekursionstiefe (der Zahl der Verschachtelungen) ändern müssen, damit die Abbruchbedingung anhand veränderter Parameter erkannt werden kann. Eine Routine darf sich daher niemals mit den gleichen Parametern selbst aufrufen, mit denen sie aufgerufen wurde.

Im Schneider-Basic sind sämtliche Variablen, die man verwenden kann, global, da alle Unterprogramme auf sie in gleicher Weise zugreifen können. Rekursion in Basic ist nur unter sehr eingeschränkten Bedingungen möglich, da man sich einen Software-Stack (in einem Array) aufbauen muß.

In Assembler steht dem Programmierer jedoch der Hardware-Stack zur Verfügung, der Anwendung der Rekursion sind also keine Grenzen gesetzt. So wird die Rekursion auch im Basic-Interpreter und im Betriebssystem des CPC eingesetzt. Die wichtigste Anwendung ist hier die Auswertung eines Ausdrucks im Basic (siehe Abschnitt 4.5.2).

Viele Programme kommen ohne den Einsatz von Rekursion aus. Grundsätzlich läßt sich jedes Problem nicht-rekursiv lösen. In vielen Fällen ist eine rekursive Lösung jedoch eleganter als eine nicht-rekursive.

2.3.2 Transparente Ausführung von Routinen

In Abbildung 2.5.1 ist die Struktur eines verschachtelten gewöhnlichen Unterprogrammaufrufs dargestellt. Jedes Unterprogramm kehrt ins aufrufende Programm zurück, und zwar an die Stelle nach seinem Aufruf. Dieses Zurückkehren wird bekanntermaßen durch eine Rückkehradresse auf dem Stack realisiert.

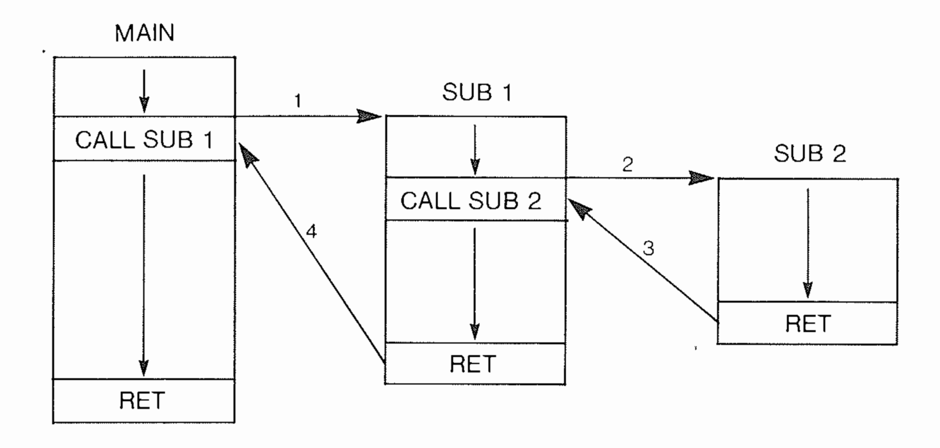
Im Schneider-CPC wird dieser klar strukturierte Programmablauf jedoch an einigen Stellen durchbrochen. Das geschieht mit dem Zweck, eine Routine transparent auszuführen. Dies ist z.B. im Basic nötig, wenn bestimmte Operationen auf dem Bildschirm ausgeführt werden sollen. Vor Ausführung einer solchen Operation setzt das Basic in der Regel eine übergebene Streamnummer (z.B. beim PRINT#-Befehl). Dann wird die eigentliche Operation in einer Routine ausgeführt und anschließend die alte Streamnummer wieder zurückgesetzt.

Dieses Setzen und Rücksetzen der Streamnummer führt eine eigene Routine (in Abbildung 2.5.2 TRANS genannt) durch, die von der eigentlichen Bildschirm-Routine (SUB in der Abbildung) transparent aufgerufen wird.

Schauen wir uns den Ablauf einmal im einzelnen an. Zuerst ruft MAIN mit einem normalen Unterprogrammaufruf die Routine SUB auf (Pfeil 1). SUB ruft dann ihrerseits die Routine TRANS mit einem normalen CALL-Befehl auf (Pfeil 2). TRANS kehrt jedoch nicht auf gewöhnlichem Weg zu SUB zurück, sondern holt die SUB-Rücksprungadresse vom Stack, um SUB dann weiterzuführen (Pfeil 3). Dadurch ist es der Routine TRANS möglich, nach der Beendigung von SUB (Pfeil 4) noch Operationen durchzuführen. In unserem Beispiel der Bildschirmoperation, die SUB ausführt, schaltet TRANS vor der Weiterführung von SUB eine bestimmte Streamnummer ein und stellt nach der Weiterführung von SUB wieder die alte Streamnummer ein. Die Routine TRANS wird also insofern transparent aufgerufen, als daß die Routine SUB von den Operationen, die TRANS nach der Weiterführung von SUB ausführt, nichts erfährt. Es wird durch diese Technik ein Aufruf am Ende von SUB gespart.

Generell führt die Manipulation mit Rückkehradressen auf dem Stack zu unübersichtlichen Programmstrukturen. Die Beschreibung des transparenten Unterprogrammaufrufs im CPC soll Sie also nicht ermuntern, dieses Problem auf gleiche Art und Weise zu lösen. Als Alternative zu der im CPC benutzten Technik wäre denkbar, der TRANS-Routine in einem Register eine Adresse zu übergeben, die dann angesprungen wird.

1. Einfacher Unterprogrammaufruf



2. Transparenter Unterprogrammaufruf

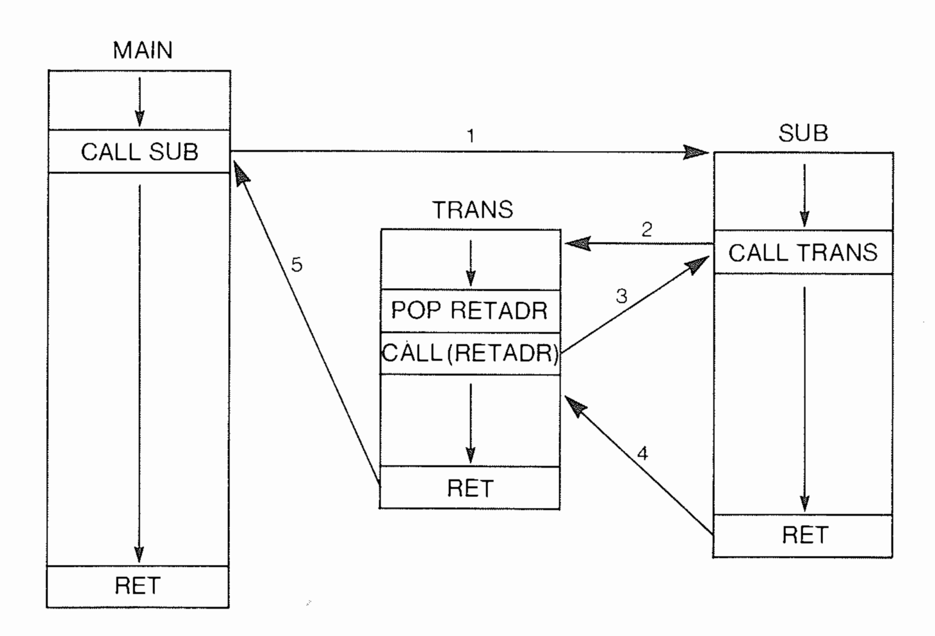


Abbildung 2.5: *Transparente Ausführung von Routinen*

2.3.3 Position Independence (Ortsunabhängigkeit)

Die meisten Maschinenprogramme, die auf dem Z80A laufen, sind für feste Adressen geschrieben. Dies rührt von den festen Adressen bei Sprüngen mit JP und CALL her. Wenn man ein ortsabhängiges Programm an eine andere Stelle im Speicher schieben würde, so wären unkontrollierte Abläufe die Folge.

Für viele Zwecke ist jedoch eine Ortsunabhängigkeit unerläßlich. Will man in ein Basic-Programm eine Zeile einfügen, so muß ein Teil des Programms verschoben werden. Dies funktioniert jedoch nur, wenn das Basic-Programm an keine feste Adresse gebunden ist. Deshalb werden bei der Einbindung einer Programmzeile in die verkettete Liste der Zeilen keine Kettungsadressen, sondern Kettungsoffsets relativ zum Zeilenanfang benutzt. In diesem Fall also die Längen der Zeilen. Ebenso werden keine Variablenadressen, sondern Variablenoffsets relativ zum Start des Variablenbereichs ins Programm gespeichert. Damit kann der Variablenbereich ebenfalls beliebig verschoben werden.

Neben der Verwendung von relativen Adressen (Offsets) gibt es noch andere Möglichkeiten, eine Ortsunabhängigkeit zu gewährleisten. Bei einem Offset folgt die Ortsunabhängigkeit aus der Allgemeinheit der Basisadresse, auf die sich die Offsets beziehen. Bei bestimmten Anwendungen ist ein Offset jedoch nicht nötig. Die allgemeine Adresse kann direkt verwendet werden. Dies wird im Schneider-Basic beispielsweise bei der Ausführung von Arithmetik-Routinen praktiziert: Eine Routinenadresse wird in einem Register an ein Unterprogramm übergeben, das die Argumente der Arithmetik-Funktion umformt und dann die gewünschte Routine aufruft.

Die Vorraussetzung für die Ortsunabhängigkeit der Betriebssystem-Routinen sind Sprungvektorentabellen im RAM (siehe Kapitel 3.3). Da die Sprungadressen der RAM-Vektoren beliebig verändert werden können, brauchen die Betriebssystem-Routinen nicht an festen Adressen zu liegen. Das machte die Änderung der 664- und 6128-Routinen gegenüber dem CPC 464 möglich, ohne daß die Ansprünge der RAM-Vektoren (die ja an einer festen Stelle im RAM liegen), in allen Programmen geändert werden mußten.

3 Beschreibung des OPERATING SYSTEMS (OS)

Das Betriebssystem (Operating System) stellt einen Teil der im CPC implementierten Firmware dar; das Basic bildet den anderen Teil. Seine Aufgaben sind vor allem, die peripheren Einheiten des Systems (also besonders Geräte zur Ein- und Ausgabe von Daten, wie z.B. die Tastatur, den Drukker, den Bildschirm usw.) zu unterstützen. Jeder Einheit ist dabei ein Programmteil des Betriebssystems zugeordnet, der dazu dient, diese Einheit zu verwalten. Ziel dabei ist es, die Features, die das Gerät von sich aus hat, möglichst gut zu nutzen, so daß ein Anwenderprogramm ohne Schwierigkeiten mit der Peripherie kommunizieren kann. Wichtig ist dabei auch, daß das Anwenderprogamm über die Art des Gerätes bzw. dessen Aufbau wenn möglich nicht informiert zu sein braucht. Das ist die unterste Stufe der Geräteunabhängigkeit (Device Independence). In diesem Zusammenhang betrachten wir auch das Basic als ein Benutzerprogramm des Operating Systems. Aber auch jedes selbstgeschriebene (Maschinen-)Programm ist ein solches Benutzerprogramm.

Diese Zuordnung von ganzen Programmsegmenten zu bestimmten Geräten (bzw. allgemeiner: bestimmten Aufgaben, da z.B. das Kernel im eigentlichen Sinne nicht mit einem Gerät kommuniziert) macht eine Unterteilung des gesamten Betriebssystems in diese semantischen Blöcke sinnvoll, die auch im Speicher geschlossen zusammen liegen. Das Firmware Manual zum CPC-Operating System nennt diese Blöcke "Packs", Grund genug für uns, uns dieser Nomenklatur anzuschließen. Im Folgenden werden alle Packs ausführlich beschrieben.

3.1 Das KERNEL (KL)

3.1.1 Allgemeines

Das Kernel stellt - zusammen mit dem Machine Pack - die unterste Ebene des Betriebssystems dar. Während das Machine Pack jedoch die Schnittstelle der CPC-Hardware zur Software darstellt, und man es daher als unterste physikalische Ebene verstehen kann, könnte man dem Kernel die Stellung der untersten logischen Ebene einräumen. Seine Aufgaben sind nicht so sehr hardwareorientiert, sondern es hat viel eher mit den grundlegenden Strukturen, in denen die Software im Schneider-Computer organisiert ist, zu tun (obschon eine gewisse Abhängigkeit von der Hardware, z.B. beim Banking für die Upper-ROMs, wohl kaum abzustreiten ist).

Die wichtigsten Aufgaben des CPC-Kernels sind daher auch die Verwaltung der verschiedenen ROMs im Bereich von $C000 bis $FFFF (sogenannte Upper- oder Hi-ROMs), die Verwaltung von Events und den damit zusammenhängenden Strukturen sowie die Koordinierung des Interrupts und die Ausführung damit verbundener Aufgaben.

3.1.2 Das Banking im CPC

Wie vermutlich bekannt, verfügt der Schneider-Computer über zwei interne ROMs, beide in einer Größe von 16 KByte. Genau genommen existiert im CPC nur ein 32-KByte-ROM, das logisch jedoch in zwei 16-KByte-Blöcke untergliedert ist. Das erste ROM, das sogenannte Lower- oder auch Lo-ROM, in dem sich das Betriebssystem des CPC befindet, belegt dabei die unteren 16 KByte der insgesamt 64 KByte, die der Z80 adressieren kann. Das Hi-ROM belegt die oberen 16 KByte. Zwischen den beiden ROMs ist also noch ein freier Platz von 32 KByte. Die zugrunde liegende Hardware wird in Abschnitt 1.3.6 erläutert und der Mechanismus mit einem Schaubild verdeutlicht.

Zusätzlich zu den ROMs ist der gesamte Speicherbereich des Z80 mit RAM ausgelegt, d.h. neben den beiden 16-KByte-ROMs verfügt er noch über insgesamt 64 KByte RAM. Der CPC 6128 hat zwar 128 KByte RAM, davon sind jedoch nur maximal 64 KByte gleichzeitig ansprechbar. Die Verwaltung der verschiedenen RAM-Banken wird vom Kernel kaum unterstützt (siehe Abschnitt 3.1.3). Daher betrachten wir hier nur die eingeschalteten 64 KByte. Näheres zum RAM-Banking finden Sie in Kapitel 1.3.7.

Dort, wo die ROMs liegen (in den oberen bzw. unteren 16 KByte), liegt also auch noch RAM-Speicher. Wenn der Prozessor eine Adresse auslesen möchte und die entsprechende Adresse auf den Adreßbus legt, liefern ihm sowohl RAM als auch ROM ein Byte auf dem Datenbus. Das kann natürlich nicht gutgehen, deshalb wurde bereits in der Hardware des CPC eine entsprechende Vorkehrung getroffen, die es ausschließt, daß sich in den "kritischen" Bereichen zwei verschiedene Speicher angesprochen fühlen. Der Prozessor kann über einige Bits im Gate Array (das ist *der* Steuerchip im Schneider-Computer, der alle wesentlichen Abläufe kontrolliert, u.a. auch die Speicherzugriffe des Prozessors) bestimmen, welchen Speicher er in den unteren und welchen er in den oberen 16 KByte ansprechen möchte. Die dafür vorgesehenen Bits befinden sich im Kontroll-Register des Gate Arrays.

Diese Technik des An- und Abschaltens von Speicherbanken nennt man "Banking". Es dient dazu, den adressierbaren Speicherbereich von Mikro-

prozessoren zu vergrößern. Das trifft im besonderen auf 8-Bit-CPUs zu, die einen relativ begrenzten Adreßraum von meist nur 64 KByte haben, entsprechend ihrer Adreßbusgröße von 16 Bit. Im Gegensatz stehen dazu die neueren 16-Bit CPUs, die teilweise über einen direkten Adreßraum von 1 MByte, also 1024 KByte und mehr verfügen. Im CPC 6128 z.B. sind Speicher mit insgesamt 176 KByte (128 KByte RAM, 48 KByte ROM) eingebaut, obwohl nur 64 KByte vom Z80 direkt angessprochen werden können. Diese Technik des Banking ermöglicht den älteren Prozessoren (zu denen ohne Zweifel auch der Z80 gehört) in der KByte-Schlacht, die gegenwärtig geführt wird, noch eine Weile konkurrenzfähig zu bleiben. Doch trotz aller Vorteile, die durch die Technik des Banking zu erzielen sind, gibt es gegenüber der direkten Adressierung auch einige Nachteile: Durch das Umschalten auf jeweils andere Speicherbanken geht Zeit verloren, die, wenn öfter umgeschaltet werden muß, schon ziemlich ins Gewicht fällt. Außerdem ist die Programmentwicklung schwieriger, da der Programmierer das - in einigen Computern schon recht komplexe - Banking mit in seine Überlegungen einbeziehen muß.

Die Konstrukteure des Schneider-Computers haben es im Zuge der Speicherplatz-Erweiterung nicht bei einem einfachen Banking belassen. Denn um nun zusätzliche Software (vor allem solche, die wie das DOS praktisch immer benötigt wird) nicht in den mit 48 KByte recht begrenzten freien RAM des CPC laden zu müssen, ließen sie sich gleich ein mehrfaches Banking für das Upper-ROM einfallen. Das Verfahren ist im Prinzip recht einfach: Es gibt nicht nur ein oberes ROM, sondern mehrere (im Schneider CPC genau 252), die voneinander durch eine Nummer zwischen 0 und 251 ($00 bis $FB) zu unterscheiden sind (siehe Abschnitt 1.3.6). Wenn nun ein Programm auf ein bestimmtes oberes ROM zugreifen möchte, so muß es nicht nur das obere ROM einschalten, sondern auch noch die Nummer des ROMs an eine Logik, die dann das gewünschte ROM auswählt, übergeben. Diese Logik muß im Schneider extern an den Expansion Bus angeschlossen werden. (Der Kasten mit dem Floppy-ROM z.B. enthält eine solche Logik. Die Nummer des Floppy-ROMs ist 7, die Nummer des immer vorhandenen Basic-ROMs ist 0.) Wichtig ist zu bemerken, daß diese Technik nur für das obere ROM implementiert werden kann. Für das untere ROM gilt nach wie vor das Prinzip des einfachen Bankings.

Wie Sie vermutlich schon bemerkt haben, erweitert diese Möglichkeit, 251 zusätzliche 16-KByte-ROMs anzuschließen, den adressierbaren Speicher um ein Vielfaches. Dennoch gibt es Nachteile: das Banking-System wird dadurch noch komplexer, das Ansprechen eines ROMs noch schwieriger und zeitaufwendiger. Vor allem aber ist die Nutzbarkeit dieses zusätzlichen Adreßraumes aus Hardware-Gründen auf ROMs beschränkt. Weiterhin ist 251 zwar eine schöne Zahl, trotzdem ist diese Möglichkeit, derart viele

ROMs anzuschließen, sicherlich reiner Overkill. Dennoch: Systeme, wie z.B. das der Floppy, verdanken ihren relativ niedrigen Anspruch an RAM ausschließlich dieser Technik.

3.1.3 Banking und RSX im Kernel

Nachdem wir uns nun eingehend mit dem Banking im Schneider-Computer befaßt haben, kommen wir jetzt zu der Rolle, die das Betriebssystem dabei spielt. Wir hatten schon erwähnt, daß der Programmierer normalerweise einen recht hohen Programmaufwand treiben muß, um das Banking in seinem Programm zu implementieren, und wir hatten dies als einen der Hauptnachteile des Bankings hingestellt. Im CPC jedoch nimmt das Betriebssystem (bzw. das Kernel) dem Programmierer die Sorge darum ab. Das Kernel verfügt über eine ganze Reihe von Routinen, die die einzelnen Speicherbanken verwalten und den User (bzw. das Basic) sehr komfortabel beim Ansprechen der verschiedenen ROMs unterstützen. So gibt es z.B. die Routinen KL U ROM ENABLE und KL L ROM ENABLE zum Einschalten des oberen bzw. des unteren ROMs, analog dazu die Routinen KL U ROM DISABLE und KL L ROM DISABLE zum Ausschalten der ROMs. Um ein bestimmtes oberes ROM auszuwählen gibt es KL ROM SELECT, und um den alten Zustand (d.h. das alte obere ROM) wiederherzustellen KL ROM DESELECT bzw. KL ROM RESTORE, wenn lediglich die alten ROM-Switches wieder eingeschaltet werden sollen, ohne daß das alte obere ROM wieder eingesetzt wird. Diese Routinen verwalten das Banking auf der untersten Ebene.

Beim CPC 6128 werden zusätzlich zum ROM-Banking und unabhängig davon noch die acht 16-KByte-RAM-Banken verwaltet (siehe Abschnitt 1.3.7). Hierzu stellt das Kernel die Routine KL RAM SELECT zur Verfügung, die die vom Benutzer übergebene RAM-Konfiguration in das Gate Array schreibt. Der Routinen-Name stammt im Gegensatz zu den anderen Namen nicht aus dem Firmware Manual, da uns ein solches für den 6128 nicht zur Verfügung stand.

Von den 252 möglichen ROMs spielen die sieben ROMs mit den Nummern von 1 bis 7 eine besondere Rolle. Beim CPC 664/6128 haben die ROMs 0 bis 15 diese Sonderstellung. Sie werden nämlich von einer Routine namens KL ROM WALK nacheinander durchgegangen und - eine bestimmte Kennung vorausgesetzt - in eine verkettete Liste eingetragen, die die momentan vorhandenen RSX-Erweiterungen enthält (bzw. die Nummern der Erweiterungs-ROMs). Diese verkettete Liste spielt eine Rolle beim Auffinden von RSX-Kommandos. RSX steht hierbei für *R*esident *S*ystem e*X*tension, also eine immer vorhandene Erweiterung des Systems (wie z.B. der Floppy-

Controller). RSX-Strings sind nun Befehlswörter (ganz ähnlich den Basic-Befehlswörtern), die für eine solche Erweiterung reserviert sind.

Von Basic aus spricht man einen RSX-Befehl dadurch an, daß man dem Befehlswort einen vertikalen Strich voranstellt (siehe Abschnitt 4.8.2). Der Basic-Interpreter ruft dann die Kernel-Routine KL FIND COMMAND auf. Sie hat die Aufgabe, den RSX-Befehl in den eingeschalteten Erweiterungs-ROMs (oder auch in einer RAM-RSX-Erweiterung) zu suchen und die Adresse des entsprechenden Befehlsroutine zurückzugeben, wenn das ROM, in dem der Befehl gefunden wurde, in der verketten Liste der RSX-Erweiterungen steht, bzw. den Befehl gleich auszuführen, wenn er in einem der anderen ROMs gefunden wurde.

Diese Möglichkeit der RSX-Erweiterung wird z.B. beim Einbau der Floppy-Kommandos in das normale Basic benutzt. Einige Routinen des Kernels haben damit zu tun: KL FIND COMMAND sucht, wie der Name sagt, sie nach einem RSX-String. KL ROM WALK geht die unteren 7 bzw. 16 ROMs durch und hängt sie gegebenenfalls in die Linked List ein. KL INIT BACK initialisiert ein "RSX-Hintergrund-ROM" (background ROM) indem es testet, ob es ein Hintergrund-ROM ist und es gegebenenfalls in die verkettete Liste der RSX-Erweiterungen einträgt. Der Name Hintergrund-ROM rührt daher, daß dieses ROM für das übergeordnete System, also normalerweise Basic, nicht "sichtbar" ist. Man nennt dies auch transparent, da es immer nur für die Dauer eines Befehls eingeschaltet bleibt und danach wieder ausgeschaltet wird. Im Gegensatz dazu bleibt ein Vordergrund-ROM, also z.B. das Basic, über längere Zeit eingeschaltet, weil es ein eigenständiges System darstellt. Schließlich hängt KL LOG EXT eine neue RSX-Erweiterung in die VL der RSX-Erweiterungen ein. Der VL-Eintrag, der der Routine übergeben werden muß, besteht entweder aus einer ROM-Nummer (kleiner $FC) oder aus einer Adresse, bei der die RSX-Erweiterung im RAM liegt. Das Hi-Byte dieser Adresse darf nicht null sein, da die Adresse sonst als ROM- Nummer interpretiert wird.

Wie schreibt man nun selbst eine RSX-Erweiterung? Die RSX-Befehlswörter und RSX-Einsprungadressen müssen in Tabellen zusammengefaßt werden. Die zwei Byte vor der Einsprung-Tabelle werden als Zeiger auf die Tabelle der Befehlswörter interpretiert. Zwei Byte später folgen die Einsprünge für die einzelnen Befehle, und zwar im Abstand von jeweils drei Byte. Am einfachsten ist es, hier JP-Befehle zu den einzelnen Befehlsroutinen einzutragen. Die Befehlswörter werden ebenfalls hintereinander in eine Tabelle eingetragen. Das Ende eines Befehlsworts wird durch ein gesetztes 7. Bit des letzten Bytes und das Ende der Befehlswort-Tabelle durch eine Null gekennzeichnet. Die Befehlswort-Tabelle bestimmt somit die Anzahl der RSX-Befehle in einer Erweiterung und damit auch die Länge der

Einsprung-Tabelle. Einen Zeiger vor letztere Tabelle (genau zwei Byte vor dem ersten Einsprung) muß man nun der Routine KL LOG EXT übergeben, damit die Erweiterung korrekt eingehängt wird. Ist die Erweiterung in einem externen ROM, so wird die ROM-Nummer übergeben. Die Einsprung-Tabelle beginnt dann bei $C006, der Zeiger auf die Befehlswort-Tabelle steht also bei $C004. Weitere Informationen über RSX erhalten sie in Abschnitt 4.8.2.

3.1.4 Die Bearbeitung von Events

Die Bearbeitung von Events und deren Verwaltung gehören zu den zentralen Aufgaben des Kernels. Um nun im Folgenden darauf näher eingehen zu können, ist es zunächst einmal unerläßlich, den Begriff des "Events" näher zu erläutern.

3.1.4.1 Der Begriff Event

Ein Event (deutsch etwa "Ereignis") ist zunächst einmal nichts weiter als eine Routine, die bei bestimmten Gelegenheiten aufgerufen wird. Je nach Art dieser Gelegenheit unterscheidet man zwei Arten von Events:

1. **Synchrone Events:** Sämtliche synchrone Events, die im Moment vom Kernel verwaltet werden, sind in einer verketteten Liste, der "Synchronous Pending Queue" (SPQ) eingetragen (geordnet nach einer Priorität, auf die wir noch zu sprechen kommen werden). Alle Events in dieser Liste warten auf ihre Ausführung, man sagt, sie seien "schwebend" (pending). Die Einträge in dieser Liste sind "Event Blocks", deren Herzstück ein Zeiger auf die eigentliche Event-Routine ist. Sie wird angesprungen, wenn der Event zur Ausführung kommt. Ein solcher Event Block enthält daneben noch weitere Informationen, z.B. eine Koppeladresse, einen Zeiger also auf den nächsten Eintrag in der Liste. Auf den genauen Aufbau eines Event Blocks wird noch später eingegangen werden. Die Events in der SPQ werden nacheinander abgearbeitet, und zwar nur dann, wenn das kontrollierende Programm (im allgemeinen der Basic-Interpreter) dem Kernel sagt, daß es jetzt den nächsten synchronen Event auszuführen hat. Daher kommt auch der Name "synchron" für diese Art der Events, da sie immer synchron zum Programm abgearbeitet werden. Ein typischer synchroner Event ist z.B. der Break-Event, der von Basic aus benutzt wird. Das Basic springt jedesmal bei Eintritt in die Interpreterschleife die Routine im Kernel an, die den nächsten synchronen Event in der SPQ ausführt, wenn einer vorhanden ist. Der Break-Event, der vom Keyboard Manager in die SPQ eingehängt wird, darf auch nur an dieser Stelle die Behandlung eines Breaks auslösen: nur zwischen zwei Basic-Befehlen und nicht während ein Basic-Befehl gerade ausgeführt wird.

2. Asynchrone Events: Der Event Block von asynchronen Events hat genau den gleichen Aufbau wie der von synchronen Events (mit der Ausnahme eines Flags, das den Event-Typ beschreibt). Desweiteren werden *auch* asynchrone Events in einer verketteten Liste zusammengefaßt, allerdings nicht in der SPQ, sondern in einer eigenen "Asynchronous Pending Queue" (APQ) oder auch "Interrupt Pending Queue" (IPQ) genannt. Der große Unterschied zu den synchronen Events besteht in dem Anlaß, der die asynchronen Events zur Ausführung bringt. Werden die synchronen Events nach und nach durch den Ansprung einer entsprechenden Routine durch das kontrollierende Programm ausgeführt, so wird die gesamte APQ innerhalb eines Interrupts bearbeitet. Es erklärt sich somit auch der Name "asynchron": da ein Interrupt ohne Rücksicht auf das gerade ablaufende Programm auftritt, werden auch die asynchronen Events ohne Rücksicht darauf, wo sich das Basic z.B. im Augenblick befindet, ausgeführt - d.h. auch mitten in einem Befehl. Das schränkt auf der einen Seite natürlich die Verwendbarkeit asynchroner Events ein, da diese z.B. keine Speicherstellen im Systembereich des Basics ändern dürfen (was mitten in der Befehlsausführung sicherlich verheerende Folgen hätte). Das öffnet aber auf der anderen Seite ganz neue Möglichkeiten, Echtzeit-Systeme zu realisieren, da der Interrupt streng periodisch mit einer Frequenz von etwa 300 Hz auftritt. Mit synchronen Events ist das kaum möglich.

Fassen wir die vielleicht ein wenig verwirrenden Erkenntnisse, die wir soeben gewonnen haben, noch einmal kurz zusammen: Ein Event ist eine Routine, deren Adresse in einem zugehörigen Event Block eingetragen ist (zusammen mit einigen anderen Parametern, wie z.B. der ROM-Konfiguration beim Aufruf des Events). Der Event Block wird vom Betriebssystem des Schneider Computers derart verwaltet, daß die Blöcke in "Pending Queues" eingereiht werden, die die Form einer verketteten Liste haben. Es gibt nun zwei verschiedene Arten von Events, wobei es für jede Klasse genau eine Pending Queue gibt: synchrone Events und asynchrone Events (entsprechend gibt es eine Synchrone Pending Queue, SPQ, und eine asynchrone Pending Queue, APQ oder IPQ). Die Abarbeitung der SPQ (und damit der Aufruf der synchronen Events) geschieht ausschließlich auf den Befehl des kontrollierenden Programms, während die gesamte APQ innerhalb eines Interrupts abgearbeitet wird (welcher mit einer Frequenz von 300 Hz auftritt). Es ist wichtig, diese grundlegenden Strukturen verstanden zu haben, wenn man die Event-Bearbeitung im Schneider-Computer verstehen möchte. Also lesen Sie sich diesen Abschnitt ruhig noch einmal durch.

3.1.4.2 Chains und ihre Bedeutung

Nachdem wir oben die Begriffe Event und Pending Queue (PQ) geklärt haben, kommen wir jetzt zu einer weiteren Struktur, die unter die Event-Verwaltung fällt, zu den "Chains" (Ketten).

Wie im letzten Abschnitt vielleicht schon durchgeklungen ist, werden die Queues, die ja Event für Event abgearbeitet werden, während der Bearbeitung immer kürzer. Ein Event wird nach der Ausführung wurde, aus der Queue ausgehängt. Wenn man aber z.B. eine Routine immer wieder ausgeführt haben möchte, so ist folglich das Einhängen in eine PQ keine Lösung für das Problem, da die Routine daraufhin genau einmal ausgeführt wird und dann aus der PQ verschwindet. Man könnte dann die Routine genauso gut auch als Unterprogramm direkt aufrufen, anstatt sie mühsam in eine verkettete Liste einzuhängen, sie irgendwann einmal vom Betriebssystem ausführen und dann wieder aushängen zu lassen. Was man also benötigt, ist eine weitere Liste, in der alle Events eingetragen sind, die man periodisch in die PQ einhängen möchte. In dieser Liste werden die Events dann z.B. bei jedem Interrupt durchgegangen und in die entsprechende PQ eingehängt, ohne die Events dabei aus dieser Liste auszuhängen. Alle Events verbleiben somit in dieser Liste, wenn der Programmierer es nicht anders bestimmt.

Eine solche Liste, die dazu dient, periodisch zu wiederholende Events aufzuheben, nennt man im Schneider-Computer eine "Chain". Im CPC gibt es gleich drei verschiedene Chains, die sich ganz ähnlich, wie auch die Pending Queues, durch die Anlässe unterscheiden, zu denen sie durchgegegangen und Event für Event ihrer PQ zugewiesen werden. Man nennt diesen Vorgang, einen Event aus einer Chain in eine der beiden PQs einzuhängen, auch "kicken". Die erste Chain ist die "Fast Ticker Chain" (FTC). Sie ist eigentlich die Chain, die wir oben als Beispiel angeführt haben: Die Fast Ticker Chain wird bei jedem Interrupt durchgegangen, d.h. bearbeitet. Die zweite Chain ist die "Ticker Chain". Sie wird nicht mit 300 Hz bearbeitet (wie die FTC), sondern nur mit etwa 50 Hz, nämlich bei jedem sechsten Interrupt. Schließlich ist da noch die "Frame Fly Chain", die nur bearbeitet wird, wenn der Video-Chip im Schneider-Computer eine vertikale Synchronisation (VSYNC) ausgibt, d.h. wenn das (Fernseh-/Monitor-)Bild aufgebaut ist und der Elektronenstrahl von der rechten unteren Ecke in die linke obere "zurückfährt".

Die Struktur einer Chain ist ganz ähnlich der einer Pending Queue: sie ist eine verkette Liste, wobei die Koppeladresse auf den nächsten Listeneintrag, dem eigentlichen Event Block, einfach vorangestellt wird. Bei Einträgen in der Ticker Chain kommen noch einige Zähler davor. Man kann auch sagen, der Event Block erhält einen "Kopf" für die jeweilige Chain.

3.1.4.3 Der Aufbau von Event Blocks

In den bisherigen Abschnitten über die Bearbeitung von Events durch das Schneider-Operating-System haben wir die (informatischen) Strukturen kennengelernt, die für die Verwaltung der Events nötig sind. Dieser Abschnitt soll nun eher der "Praxis" in dem Sinne gewidmet sein, daß wir endlich dazu kommen, uns über den Aufbau eines Event Blocks Gedanken zu machen. Dies ist vor allem für jemanden wichtig, der Events in eigenen Programmen zu nutzen gedenkt. Wenn Sie lediglich das Prinzip interessiert, nach dem Events funktionieren, so kann das Folgende nur noch zur Verdeutlichung beitragen.

Hier folgt jetzt eine kurze Übersicht über die Bedeutung der Bytes in einem Event Block. Eine Erklärung bisher nicht erklärter Begriffe wurde hinten angestellt.

Byte 0/1:	Koppeladresse in der Pending Queue (KAPQ)
Byte 2:	Pending Queue Zähler
Byte 3:	Priority Byte oder auch Class Byte.

Die einzelnen Bits haben folgende Bedeutung:

Bit 0:	=0:	Sprungadresse ist eine Far Address
	=1:	Sprungadresse ist eine Near Address
Bit 1-4:		Priorität eines Synchronous Events
Bit 5:	=0:	normaler Zustand für sync. Event eingeschaltet
	=1:	Synchronous Pending Queue wird "eingefroren"
Bit 6:	=0:	normaler Event
	=1:	Express
		async.: wird sofort ausgeführt
		sychron.: kann nicht gesperrt werden (Priorität)
Bit 7:	=0:	Synchronous Event
	=1:	Asynchronous Event
Byte 4/5:		Adresse der Event-Routine
Byte 6:		ROM-Konfiguration, nur für Far Call

Die Koppeladresse in der Pending Queue ist ein Zeiger auf den nächsten Eintrag in der entsprechenden PQ. Sie erinnern sich sicher, daß die PQ die Struktur einer verketteten Liste hatte. Da die Pending Queues ausschließlich vom Betriebssystem verwaltet werden, ist die Koppeladresse kein vom Benutzer manipulierbarer Parameter und für ihn eigentlich auch kaum von Interesse.

Ist b7 gleich 0, so gibt der PQ-Zähler an, wie oft die Event-Routine ausgeführt werden soll. Ist b7 gleich 1, so wird der Event nicht eingehängt, wenn der KL EVENT übergeben wird - er ist also ausgeschaltet. (Durch einen Fehler im ROM funktioniert der PQ-Zähler nicht in der oben beschriebenen Form. Ein Event wird nur eingehängt, wenn sein Zähler gleich 0 ist, und dann auch nur genau einmal ausgeführt. Der Zähler ist

somit kein Zähler, sondern nur mehr ein Flag, das bestimmt, ob der Event eingehängt werden soll, oder nicht.)

Eine Near Address liegt im RAM von $4000 bis $BFFF (im "zentralen RAM" zwischen den beiden ROMs) bzw. im unteren ROM. Eine Far Address liegt im oberen ROM. Je nach Event Routine muß der Benutzer entscheiden, welcher Art die Adresse der Routine ist. Ist es eine Far Address, so muß er zusammen mit der Routinen-Adresse auch noch die entsprechende ROM-Konfiguration festlegen (in Byte 6).

Hinter dem Event Block (also ab Byte 7) beginnt die "User Area" (Benutzerfeld). Sie kann beliebig lang sein und auch beliebige Daten enthalten, da sie vom Kernel nicht mehr verwaltet und kontrolliert wird. Sie kann jedoch von der Event Routine besonders komfortabel genutzt werden, da beim Aufruf einer Near Address das Kernel einen Zeiger auf zwei Bytes vor das Benutzerfeld in DE hinterläßt. Bei einer Far Address zeigt der Zeiger drei Byte vor das Benutzerfeld und steht nicht in DE, sondern in HL.

Bei allen Event Blocks, die der Routine KL EVENT nicht direkt übergeben, sondern während eines Interrupts aus einer der drei Chains heraus gekickt werden, kommt zu den oben angeführten sieben Bytes des eigentlichen Event Blocks noch ein sogenannter Kopf für die jeweilige Chain. Für die Fast Ticker Chain und die Frame Fly Chain sind die Köpfe identisch: sie bestehen lediglich aus einer Koppeladresse für die Verkettung der Events innerhalb der Chain (die, wie wir ja mittlerweile wissen, wie die Pending Queues von ihrer Struktur her verkettete Listen sind). Da auch dieses Feld (ebenso wie die PQ-Kettung) vom Betriebssystem versorgt wird, hat der User lediglich dafür Sorge zu tragen, daß die beiden Bytes für die Chain-Kettung vor dem Event-Block auch nicht anderweitig benutzt werden, also frei sind. Alles andere erledigt nach dem Einhängen in die Queue das Kernel.

Der Kopf für einen Ticker Chain-Eintrag jedoch unterscheidet sich von den beiden anderen. Innerhalb eines Ticker-Kopfes existiert noch ein Zähler, mit dem der Benutzer festlegen kann, in welchen zeitlichen Abständen der entsprechende Event gekickt (d.h. in seine Pending Queue eingehängt) werden soll. Steht in diesem 2-Byte-Wert (der folglich Werte von 0 bis 65535 enthalten kann) z.B. die Dezimalzahl 10000, so wird der Event erst nach dem 10000sten Ticker-Durchlauf eingehängt. Da die Ticker-Chain mit einer Frequenz von 50 Hz durchgegangen wird, bedeutet dies, daß der Event erst nach 200 Sekunden gekickt wird. Zusätzlich gibt es noch einen "Reload Count", also den Zählerwert, auf den der Zähler gesetzt wird, wenn er einmal heruntergezählt wurde. Dieser Reload Count

wird natürlich nicht vom Betriebssystem verändert. Im Folgenden kurz der Aufbau des Ticker-Kopfes im Überblick:

Byte 0/1: Koppeladresse für die Ticker Chain
Byte 2/3: Tick Count, wird heruntergezählt bis auf Null, dann wird der Event gekickt und der Tick Count aus dem Reload Count neu geladen.
Byte 4/5: Reload Count, bestimmt den neuen Startwert, mit dem der Tick Count geladen wird, nachdem er einmal auf Null gezählt wurde.

Wenn Ihnen übrigens die zuletzt behandelten Zusammenhänge bekannt vorkommen, so ist dies kein Wunder: die Basic-Befehle AFTER und EVERY benutzen nämlich genau diesen Mechanismus. Das Zeitintervall, das Sie diesen Befehlen zu übergeben haben, wird direkt als Tick Count gesetzt. Beim Befehl AFTER wird der Reload Count auf Null gesetzt, als Zeichen für nur ein einziges Kicken des Events. Beim Befehl EVERY wird dagegen der Reload Count gleich dem übergebenen Wert gesetzt (und damit gleich dem ersten Tick Count). Hier ist das Schneider Basic zu loben, das diese Möglichkeit der Event-Behandlung im Kernel auch (ebenso wie viele andere Besonderheiten des Betriebssystems, wie wir noch sehen werden), dem Basic-Benutzer zugänglich macht, wenngleich mit einigen Einschränkungen, und nicht nur Maschinensprache-Freaks vorbehält.

3.1.4.4 Routinen zur Event-Behandlung

Nachdem wir nun alle für die Verwaltung von Events wichtigen Mechanismen sowie den Aufbau von Event Blocks behandelt haben, wenden wir uns abschließend den Kernel Routinen zu, die für den Benutzer den Zugang zu diesen Mechanismen ermöglichen.

INIT EVENT ist sicherlich die einfachste Routine. Sie dient dazu, einen Event Block aufzubauen (und zwar ohne Chain-Kopf) und nimmt so dem Benutzer die Sorge um den Aufbau eines Blocks ab. Man muß ihr allerdings dennoch die erforderlichen Parameter in den Prozessor-Registern übergeben, inklusive der Adresse, an der sie den Event Block generieren soll.

NEW FRAME FLY baut (mit INIT EVENT) einen Event Block auf und hängt ihn dann in die Frame Fly Chain ein. Dies geschieht mit ADD FRAME FLY, eine Routine, die einen fertigen Block an die Frame Fly Chain anhängt. DELETE FRAME FLY hängt den Event wieder aus der Chain aus.

Für die Fast Ticker Chain existieren die analogen Routinen NEW FAST TICKER, ADD FAST TICKER und DELETE FAST TICKER. Für die Ticker Chain dagegen fehlt die Routine "New Ticker", da man ihr, zusammen mit dem Tick Count und dem Reload Count, zu viele Parameter in

den Registern übergeben müßte. Ansonsten existieren auch für die Ticker Chain ADD TICKER und DELETE TICKER.

Die Routine "Scan Events" geht bei einem Interrupt die Fast Ticker Chain durch und bearbeitet gegebenenfalls auch die anderen beiden Chains. Fast Ticker Chain und Frame Fly Chain können mit der Routine KL KICK EVENT behandelt werden. Sie geht eine Chain Event für Event durch und hängt den Event ein. Wegen des abweichenden Aufbaus des Ticker-Kopfes kann die Ticker Chain nicht mit dieser Routine bearbeitet werden. Für die Ticker Chain steht die Routine "Ticker Chain bearbeiten" zur Verfügung.

Ein einzelner Event wird mit der Routine KL EVENT gegebenenfalls in die entsprechende Pending Queue eingehängt. Diese Routine wird auch von KL KICK EVENT und "Ticker Chain bearbeiten" aufgerufen, sie kann jedoch ebenso auch direkt vom Benutzer zum direkten Kicken eines Events aufgerufen werden. Das Kicken eines Events wird durch die Routine KL DISARM EVENT verhindert (d.h. der Event ausgeschaltet). Diese Routine setzt den PQ-Zähler einfach auf $C0. Sie setzt also das b7, das ja als Flag für die De-Aktivierung eines Events gedacht war.

Wie wir ja wissen, geschieht die Abarbeitung der Synchronous Pending Queue ausschließlich unter Kontrolle des User-Programms. Es gibt daher gleich ein ganzes Paket von Routinen dafür, wobei jede einen Schritt in der Bearbeitung der SPQ darstellt. Diese Bearbeitung der SPQ wollen wir nun im Folgenden erläutern.

Im Unterschied zur APQ besitzt die SPQ eine gewisse Ordnung, und zwar sind die Events in ihr in der Reihenfolge fallender Prioritäten geordnet. Diejenigen Events mit den höchsten Prioritäten stehen weiter vorne und werden deshalb auch eher bearbeitet als die anderen. Die Priorität des laufenden Events (die laufende Priorität) und dessen Routinenadresse werden gesondert gespeichert, um zu verhindern, daß während der Ausführung eines Synchronous Events ein anderer Event mit einer niedrigeren Priorität zur Ausführung gelangt. So holt KL NEXT SYNC den nächsten Event, jedoch tut sie dies nur dann, wenn die Priorität des nächsten Events größer ist als die des laufenden. Ist sie dies, so wird von KL NEXT SYNC die neue Priorität als die laufende gesetzt und auch die neue Routinenadresse als die laufende gesetzt. Die alten Werte werden jedoch dem Benutzerprogramm übergeben. Somit hat der User die Verantwortung für die Priorität. Meist werden die gleichen Werte nach der Ausführung des Events einfach wieder gesetzt.

Ist jetzt die neue Adresse geholt worden, so wird der Event mit KL DO SYNC ausgeführt, KL DONE SYNC setzt nach beendeter Ausführung wieder die alte Priorität und die alte Routinenadresse ein. Dieser Ablauf ist

typisch für die Bearbeitung der SPQ, kann jedoch natürlich variiert werden.

Bevor man die Bearbeitung synchroner Events beginnt, sollte man KL SYNC RESET aufrufen. Diese Routine setzt eine Null als laufende Priorität, die ja immer kleiner als jede andere Priorität ist. (Bedenken Sie, daß bei synchronen Events im Priority Byte das b7 auf jeden Fall immer gesetzt ist!) Dies ist die Bedingung dafür, daß die Bearbeitung der SPQ überhaupt erst einmal begonnen werden kann.

Desweiteren kann der Benutzer mit der Routine KL DEL SYNCHRONOUS einen bestimmten synchronen Event aus der SPQ aushängen, falls er darin ist. Mit KL EVENT DISABLE kann man die SPQ einfrieren, d.h. das Holen des nächsten Events blockieren. Dies geschieht, indem b5 der laufenden Priorität gesetzt und damit größer wird, als alle erlaubten Prioritäten. Damit kann der nächste Event aus der SPQ nicht geholt werden. KL ENABLE EVENT "taut" die SPQ wieder auf, indem sie b5 der laufenden Priorität wieder zurücksetzt. Synchronous Express Events können jedoch durch das gesetzte b5 *nicht* gesperrt werden, da in ihrer Priorität b6 (als Kennzeichen für Express) gesetzt ist. Das ihre Priorität bei Vergleichen mit der Sperrpriorität natürlich immer größer.

Die Routine KL POLL SYNCHRONOUS schließlich dient dazu, festzustellen, ob es in der SPQ überhaupt einen Event gibt und ob dessen Priorität größer als die Sperrpriorität ist. Wenn ja, dann wird ein gesetztes Carry zurückgegeben.

3.1.5 Die Interrupt-Behandlung

Neben dem Banking und der Verwaltung der Events gehört die Überwachung und Koordination des Interrupts zu den Hauptaufgaben des Kernels. Bevor wir uns nun aber näher mit den Prozessen befassen, die im Schneider-Computer durch den Interrupt gesteuert werden, müssen wir zunächst den Begriff des Interrupts klären.

3.1.5.1 Der Begriff des Interrupts

Ein Interrupt (deutsch etwa "Unterbrechung") ist eigentlich nichts weiter als eine neue Art der Programmverzweigung. Um den Interrupt mit den bisher bekannten Arten logisch verknüpfen zu können, stellen wir alle hier noch einmal kurz dar. Grundsätzlich sind uns drei Arten von Programmverzweigungen bisher geläufig:

1. Der einfache SPRUNG, d.h. das Fortsetzen der Programmausführung an einer anderen Stelle durch einen entsprechenden

Prozessorbefehl (beim Z80: JP bzw. JR für relative Sprünge über einen kleinen Adreßbereich hinweg). Dieses sind die einfachsten Verzweigungen, da tatsächlich nichts weiter getan wird, als dem Programmzähler einen neuen Wert zu geben (bzw. bei JR einen Wert zum Programmzähler zu addieren).

2. Der **Unterprogrammaufruf** wird im Z80-Jargon auch CALL genannt, weil der entsprechende Prozessorbefehl genau diesen Namen trägt. Erst mit der Struktur von Unterprogrammen ist modulares Programmieren und eine gute Programmstruktur möglich; sie ist daher von größter Bedeutung. Das Besondere am CALL gegenüber dem normalen JumP ist, daß der Programmzähler nicht nur mit einem neuen Wert geladen, sondern der alte Zähler auch auf den Stack gerettet wird. Ein entsprechender Befehl, der die oberste Adresse auf den Stack in den Programmzähler lädt, macht es

   Unterprogramm beendet ist. Im Z80 ist dies der Befehl RET. Auch verschachtelte Unterprogramme ("Nesting") werden möglich, was verschiedene Unterprogramm-Ebenen zur Folge hat.

3. Obwohl er Ihnen bisher vielleicht nicht als eine solche geläufig war, gehört auch der **SYSTEM RESET** zu den Programmverzweigungen. Der Reset wird ausgelöst durch ein Low-Signal an dem dafür vorgesehenen Pin des Prozessors. Daraufhin springt dieser zur Adresse $0000, löscht also den Programmzähler. Das Programm, das hier liegt, kann - zumindest theoretisch - völlig beliebiger Art sein. Der Reset kann somit als fester Sprung angesehen werden, der durch periphere Hardware ausgelöst wird. In der Praxis tritt die Einschränkung auf, daß beim Einschalten des Systems ein Reset ausgelöst wird. Das Programm bei $0000 muß folglich das System initialisieren.

Wenn man den Reset als hardware-abhängigen Sprung verstehen kann, so kann man den Interrupt als Hardware-CALL begreifen. Es gibt nämlich auch für den Interrupt einen Prozessor-Pin gibt, an dem ein Interrupt ausgelöst werden kann. Wenn an diesen Pin durch die Hardware ein Low-Signal kommt, so rettet der Prozessor den augenblicklichen Programmzähler auf den Stack und springt nach $0038, der Interrupt-Routine. Ist die Interrupt-Behandlung beendet, so kann man mit einem ganz gewöhnlichen RET an die Stelle, an der das laufende Programm unterbrochen wurde, zurückkehren. (Meistens wird jedoch vorher noch der Interrupt wieder "erlaubt", da die Auslösung eines Interrupts immer auch mit dem

Ausschalten aller folgenden Interrupts verbunden ist. Wir kommen darauf gleich noch zurück.) Der Interrupt bleibt daher unsichtbar für das Programm, insofern er zumindest die Register des ersten Registersatzes nicht verändert (worauf man beim Programmieren eigener Interrupt-Routinen achten sollte). Man nennt dies auch "Transparenz" des Interrupts. Der Interrupt eignet sich damit besonders für asynchrone Aufgaben, d.h. Dinge, die unabhängig vom laufenden Programm erledigt werden müssen. Das sind insbesondere periodische Vorgänge, die keinerlei Aufschub dulden, wie z.B. "zeitkritische Prozesse". Dies führt dazu, daß Interrupts in vielen Fällen zur ökonomischen Kommunikation mit peripheren Einheiten genutzt werden. Im CPC wird er hauptsächlich mit Hinblick auf die Verarbeitung von Events angewendet.

Wie oben bereits erwähnt, kann der Interrupt an- und abgeschaltet werden, wozu im Befehlssatz des Z80 zwei Befehle vorgesehen sind (DI zum Abschalten, EI zum Anschalten). So ist es beispielsweise sinnvoll, während der Erledigung zeitkritischer Prozesse die Auslösung des Interrupts zu unterbinden. Die Abarbeitung der Interrupt-Routine stellt nun einmal einen kaum zu kalkulierenden Zeitfaktor dar und der Interrupt tritt darüber hinaus ja auch noch asynchron zum Programmablauf auf. Bei solchen zeitkritischen Prozessen handelt es sich zumeist um die Bedienung von Ein-/Ausgabe-Einheiten, die nicht mit Quittierungs- und Übernahme-Signalen synchronisiert sind (wie z.B. der Strobe und das Busy des Druckers), sondern zeitlich genau abgestimmt sein müssen (wie die Ausgabe auf Kassette oder an eine Floppy-Station). Wenn man solche I/O-Module selber programmiert und die Interrupt-Behandlung des CPC auch benutzen möchte, so sollte man darauf achten, daß der Interrupt nicht allzu lange ausgeschaltet bleibt (sondern nur für die tatsächlich zeitkritischen Prozesse), damit der Takt wenigstens in grober Näherung konstant bleibt (wichtig eventuell bei Ticker-Events etc.).

3.1.5.2 Die Behandlung eines Interrupts

Wie wir ja wissen, wird bei Auslösung eines Interrupts zur Routine bei $0038 verzweigt. Diese Routine macht im Schneider nichts anderes, als zur eigentlichen Behandlungs-Routine nach $B939 ins RAM zu springen. Dies hat den Vorteil, daß man als Benutzer die Interrupt-Behandlung nach eigenen Wünschen gestalten kann. Wir wollen uns nun anschauen, was diese Routine normalerweise macht. Sie wird von $03CA aus dem Lo-ROM kopiert. Für genauere Angaben beziehen Sie sich bitte auf das ROM-Listing.

Als erstes schaut die Routine nach, ob dies bereits die zweite Interrupt-Ebene (also ein Interrupt im Interrupt) ist. Ist dies der Fall, so wird der

letzte Interrupt als externer Interrupt betrachtet und entsprechend behandelt. Ist dies erst die erste Ebene, so wird dafür das CY-Flag im zweiten Registersatz gesetzt, der neben der I/O auch für den Interrupt benötigt wird. Daraufhin wird für einen kurzen Moment der Interrupt wieder zugelassen. An dieser Stelle tritt dann gegebenenfalls der externe Interrupt auf, findet ein gesetztes Carry vor und wird entsprechend abgearbeitet. Zu dem entsprechenden Programm ist anzumerken, daß das Einschalten des Interrupts durch den Befehl EI (Enable Interrupt) scheinbar vor dem Austauschen der beiden AF-Register geschieht. Wenn ein Interrupt an dieser Stelle auftritt, so würde dies dazu führen, daß die Interrupt-Routine das gesetzte Carry-Flag nicht im Interrupt-Carry vorfindet. Wie gesagt, dies scheint nur so. Tatsächlich läßt der Befehl EI infolge einer Besonderheit in der Konstruktion des Z80-Prozessors den Interrupt nicht sofort nach seiner Ausführung, sondern erst nach der Ausführung des nächsten Befehls wieder zu.

Wenn kein externer Interrupt auftrat bzw. nach Abarbeitung desselben werden die Fast Ticker Chain und eventuell auch die Frame Fly Chain durchgegangen und die Events darin gekickt. Zugleich wird bestimmt, ob es bereits an der Zeit ist, die Ticker Chain zu bearbeiten. Dafür wird ein Flag gesetzt und die eigentliche Bearbeitung findet dann erst später statt. Wenn es nach dieser FTC/FFC-Bearbeitungsroutine (Scan Events) keine Einträge in der Asynchronous Pending Queue gibt und die Ticker Chain noch nicht bearbeitet werden muß, wird die Interrupt-Routine beendet. Gleiches geschieht, wenn der Interrupt während der Bearbeitung der APQ auftrat. Andernfalls wird die APQ durchgegangen, die Ticker Chain bearbeitet und die APQ nochmals abgearbeitet, falls aus der TC asynchrone Events gekickt wurden.

Im CPC 664/6128 wurde noch eine zusätzliche Routine im Kernel implementiert: KL SCAN NEEDED. Sie dient dazu, den Ticker-Frequenzteiler auf 1 zu setzen, so daß beim nächsten Interrupt ein Ticker ausgelöst wird. Dies ist z.B. sinnvoll, wenn man sicherstellen mächte, daß die Tastatur beim nächsten Interrupt abgefragt wird (die Routine Scan Keyboard wird nur bei jedem Ticker aufgerufen). Wenn der Interrupt eine längere Zeit ausgeschaltet werden muß und hält man damit das Risiko, eine Taste zu "verpassen", möglichst niedrig.

3.1.6 Die Restart-Routinen

Neben den bekannteren Programmverzweigungen wie CALL, JP und JR gibt es beim Z80 noch eine Art, den Programmablauf zu steuern: mit Restarts (mnemonisch RST). Man kann sie als normale, unkonditionierte CALLs auf feste Sprungadressen betrachten. Jeder Restart hat dabei seine

eigene Adresse und damit auch eine eigene, ihm zugeordnete Routine. Vergleichbare Befehle sind z.B. beim 6502 der Break, BRK, beim 6809 die drei Software Interrupts, SWI, SWI2, SWI3, beim 68000er der TRAP-Befehl. Die Ansprünge der acht verschiedenen Restarts beginnen beim RAM-Anfang bei $0000 für RST 00h (oder RST 0) und liegen jeweils acht Bytes auseinander. Sie haben also genug Platz für eine winzige Routine bzw. einen Sprung in eine größere. Der letzte Restart, RST 38h (oder RST 7), springt nach $0038. Gegenüber den CALL-Befehlen haben Restarts natürlich den Nachteil, daß sie an diese festen Ansprünge gebunden sind. Sie können also das CALL nicht ersetzen. Häufig angesprungene Routinen sollte man jedoch als Restarts setzen, da die Bearbeitung eines RST-Befehls schneller ist und er weniger Speicherplatz benötigt, als ein CALL (1 Byte im Gegensatz zu 3 Bytes beim CALL). Im Schneider-Computer haben jedoch die einzelnen Restarts bereits eine feste Bedeutung, die man als Benutzer nur schwer ändern kann. Wegen des Systemaufbaus des CPC würde eine solche Änderung nur sehr schwer zu kalkulieren sein. Diese Bedeutung der RST-Routinen wollen wir im Folgenden eingehender betrachten.

RST 0: System Reset. Dieser Restart führt einen kompletten, unkonditionierten Kaltstart durch. Das gesamte System wird dabei neu initialisiert, das RAM gelöscht. Beim Einschalten des Rechners geschieht das automatisch.

RST 1: LO JUMP. Dieser Restart führt einen Sprung in die unteren 16 KByte aus, also in das Betriebssystem oder das darunter liegende RAM. Die Adresse folgt dem RST-Aufruf, und die Routine kehrt hinter die Adresse zurück. Da für den Sprung in einen 16-KByte-Bereich nur 14 Adreßbits benötigt werden, sind die Bits 14 und 15 des folgenden Words einer anderen Verwendung zugeführt: Sie geben an, ob das untere (b14) bzw. das obere (b15) ROM beim Ansprung der Routine angeschaltet sein, oder ob der entsprechende RAM-Bereich ausgewählt werden soll. Eine Null schaltet das entsprechende ROM an.

RST 2: SIDE CALL. Während der RST 1 in den Bereich von $0000 bis $3FFF springt, geht die Programmkontrolle bei einem RST 2 an eine Routine im Bereich von $C000 bis $FFFF. Sonst jedoch sind sich beide Restarts sehr ähnlich: auch dem RST 2 folgen zwei Bytes, deren untere 14 Bits den Offset der Routinenadresse bezüglich $C000 darstellen, wohingegen die oberen beiden Bits zur laufenden ROM-Nummer addiert werden. Es ist somit möglich, von einem Erweiterungs-ROM aus in eines der drei darauf folgenden ROMs zu springen, ohne daß die eigene Nummer bekannt sein muß (relative Adressierung der Extension-ROMs).

RST 3: FAR CALL. Im Gegensatz zu den beiden vorangegangenen Restarts ist man mit Hilfe des Far Calls in der Lage, einen Sprung an eine beliebige Stelle im 64-KByte-Adreßraum des Z80 auszuführen. Auch die Parametrisierung dieses Restarts unterscheidet sich von den beiden vorangegangenen: dem RST-3-Befehl folgt nicht die Adresse der Routine, vielmehr folgt ihm ein Zeiger auf einen drei Byte langen Parameterblock. Das erste Word dieses Blocks stellt dabei die Adresse der Routine dar, die man anspringen möchte, das dritte Byte ist die (ROM-)Konfiguration. Die Werte von 00 bis $FB bedeuten dabei die Nummer des Hi-ROM, das man beim Ansprung der Routine ausgewählt haben möchte. Hat die Konfiguration einen Wert von $FC bis $FF, so stellen die unteren beiden Bits die ROM/RAM-Switches für die unteren (b0) und für die oberen (b1) 16 KByte dar. Eine Null bedeutet, daß das entsprechende ROM ausgewählt ist.

RST 4: RAM LAM. Dieser Restart ermöglicht einer ROM-Routine einen einfachen (Lese-)Zugriff auf das gesamte RAM, also auch auf das darunter liegende. Die Hardware des Schneider-Computers ist derart ausgelegt, daß alle Schreiboperationen automatisch auf das RAM geleitet werden, unabhängig von den ROM/RAM-Switches im Gate Array. Die Routine RAM LAM (Load Accumulator from Memory) lädt den Akku mit dem Byte, auf das das HL-Register beim Aufruf zeigt. HL wird dabei nicht verändert.

RST 5: FIRM JUMP. Der Name dieses Restarts, der einen Sprung über die gesamten 64 KByte ausführt, bezieht sich aus der Tatsache, daß das untere ROM (der Sitz der Operating System Firmware) für die Dauer der angesprungenen Routine eingeschaltet wird, während das obere ROM unverändert bleibt. Dem RST-Befehl folgt direkt die Routinenadresse. Die Ansprünge der Arithmetik-Routinen in der Nebentabelle der Jump-Restore-Vektoren werden mit diesem Restart realisiert.

RST 6: USER. Diese Routine besteht im wesentlichen aus einer Endlos-Schleife, die bei abgeschaltetem unteren ROM auf einen Interrupt wartet, während sie die laufende Konfiguration immer wieder rettet. Sie kann genutzt werden, um auf einen externen Interrupt zu warten, der nicht mittels eines RET in die Schleife zurückkehrt. Andernfalls wäre der Aufruf der Routine sinnlos.

RST 7: INTERRUPT. Die Routine für die Behandlung eines Interrupts und die Routine für den RST 7 fallen beim Z80 zusammen. Eine genaue Beschreibung der Interrupt-Behandlung finden Sie im Abschnitt 3.1.5.

3.2 Das MACHINE PACK (MC)

3.2.1 Allgemeines

Wie schon in 3.1.1 erwähnt, bildet das Machine Pack die Schnittstelle zwischen der Software, die auf dem CPC läuft, und der Hardware des Computers. Wie das Kernel dem Programmierer die Sorge um die grundlegenden logischen Strukturen abnahm, so ist das Machine Pack dazu da, dem Benutzer den Zugriff auf die Hardware zu erleichtern, ohne daß dieser dazu über ihren genauen Aufbau informiert sein muß.

Aus dieser Aufgabe des Machine Packs folgt auch eine seiner herausragenden Eigenschaften: Es ist nur sehr wenig strukturiert und hat eher den Charakter einer Unterprogrammbibliothek, als eines in sich abgeschlossenen, zusammenhängenden Packs (wie z.B. der Keyboard Manager). Auch gibt es kaum Strukturen, über die man etwas sagen könnte. Aus diesem Grunde wollen wir uns in der Beschreibung dieses Packs auf eine Beschreibung der einzelnen Routinen beschränken.

3.2.2 Die Routinen des Machine Packs

3.2.2.1 Systemroutinen

MC START PROGRAM: Diese Routine dient dazu, ein Programm zu starten, dessen Adresse ihr übergeben wird. Das Programm soll jedoch in einem definierten System gestartet werden. Daher werden vorher sämtliche Teile des Betriebssystems initialisiert. Programme, die auf diese Weise angesprungen werden, sind zumeist keine einfachen User-Routinen, sondern eher eigenständige, unabhängige Systeme. Ein Sonderfall wird dann unterschieden, wenn die übergebene Einsprungadresse Null ist: "MC Start Program" wählt dann die Konfiguration 0 und den Ansprung $C006 aus, d.h. im Normalfall das Basic (es sei denn, das obere ROM wurde ausgewechselt).

MC BOOT PROGRAM: Diese Routine lädt ein Programm mit Hilfe einer Laderoutine, deren Adresse ihr übergeben wird. Zusätzlich werden auch noch alle wesentlichen Systemteile initialisiert (Keyboard Manager, Screen Pack und Text Screen Pack). Das geladene (gebootete) Programm wird dann mit MC Start Program gestartet. Die Startadresse muß von der Lade-Routine zurückübergeben werden.

RESET CONT'D: Dies ist die Hauptroutine, die nach einem Reset ausgeführt wird. Sie initialisiert den Video-Chip (je nach Bildwiederholfrequenz unterschiedlich) und springt dann mit MC Start Program das Basic an.

3.2.2.2 Routinen zur Bildschirmbehandlung

MC SET MODE: Mit dieser Routine wird der Bildschirmmodus eingestellt.

MC CLEAR INKS: Setzt den Wert für Border und alle anderen Farbstifte auf einen Wert.

MC SET INKS: Setzt die Farbstift-Register im Gate Array entsprechend einer Tabelle der Farben, deren Adresse ihr übergeben wird.

MC WAIT FLYBACK: Wartet auf eine vertikale Synchronisation des Elektronenstrahls des Monitors, die etwa fünfzigmal pro Sekunde eintritt.

MC SCREEN OFFSET: Übergibt die vom Benutzer gelieferten Werte für SCR BASE und SCR OFFSET an das Gate Array. SCR BASE gibt dabei den Start des Speicherbereiches an, den der CRTC abfragen soll (das sogenannte Video-RAM). SCR OFFSET zeigt innerhalb dieses Bereiches auf die Speicherstelle, bei der die Abfrage beginnen soll, d.h. auf das Byte, das die obersten linken Punkte auf dem Bildschirm repräsentiert.

3.2.2.3 Routinen für die Druckersteuerung

MC RESET PRINTER: Diese Routine setzt die Indirection, die normalerweise auf die Routine springt, die ein Zeichen an Centronics ausgibt und auf den Drucker wartet, wieder auf den Ausgangswert (d.h. auf MC Wait Printer). Es kann für den Benutzer sinnvoll sein, eine eigene Routine zum Drucken eines Zeichens an diese Stelle einzubauen, wenn er z.B. über ein besonderes Interface (z.B. RS 232) verfügt, das dann natürlich eine besondere Behandlungsroutine erfordert. Man muß dann lediglich die Indirection "umbiegen". Mit MC Reset Printer wird sie dann wieder "zurechtgebogen".

MC PRINT CHAR: Diese Routine springt die Indirection an (also normalerweise MC Wait Printer). Zusätzlich rettet es noch das BC-Register, das von MC Wait Printer verändert wird. Eine Neuerung im CPC 664/6128 gegenüber dem 464 ist eine Übersetzungstabelle (Translation Table), mit der man maximal 20 Zeichen durch jeweils ein anderes Zeichen ersetzen kann. Eine solche Tabelle muß als erstes Byte die Anzahl der Zeichenpaare enthalten, die in dieser Tabelle stehen. Dann folgen (im ASCII-Format) die Zeichenpaare, jeweils zuerst das zu ersetzende Zeichen und dann der Code, durch das es ersetzt werden soll. Ist letzterer gleich $FF, so wird das Zeichen ignoriert.

MC PRINT TRANSLATION: Diese Routine gibt es nur im 664 und im 6128. Sie dient dazu, eine neue Translation Table zu definieren, indem ihr die Adresse der Tabelle übergeben wird. Sofern diese Tabelle nur maximal 20 Zeichenpaare enthält, wird sie an eine eigens dafür vorgesehene Stelle im OS-RAM kopiert.

MC WAIT PRINTER: Druckt ein Zeichen aus und wartet vorher innerhalb bestimmter Zeitschranken, wenn der Drucker "busy" ist.

MC SEND PRINTER: Diese Routine schickt ein Zeichen an den Centronics-Port, der im allgemeinen für den Anschluß eines Druckers vorgesehen ist. Das Zeichen kann nur sieben Bits umfassen, b7 ist ständig auf null gehalten.

MC BUSY PRINTER: Schaut nach, ob der Printer bereit ist, ein Zeichen von Centronics zu übernehmen oder ob er beschäftigt (busy) ist. Es wird dann ein entsprechendes Flag zurückgegeben.

3.2.2.4 Sonstige Routinen des Machine Packs

MC SOUND REGISTER: Diese Routine erleichtert dem Benutzer den Zugriff auf die Register des PSG, die sonst nur über mehrfaches Umschalten einiger Steuersignale verfügbar werden. Man übergibt ihr die Nummer des Registers, auf das man zugreifen möchte und den Wert, den man in das Register schreiben will.

Scan Keyboard: Obwohl diese Routine keine offizielle User-Routine ist (beim 664/6128 ist sie eine Indirection), soll sie hier erwähnt werden, da sie für die Abfrage der Tastatur von zentraler Bedeutung ist. Der Benutzer übergibt ihr zwei Tabellen: eine, um die direkten Rückmeldungen aus der Tastaturmatrix abzuspeichern, eine zweite für die entsprechenden positiven Rückmeldungen - d.h. gedrückte Tasten sind als 1-Bit dargestellt, im Gegensatz zur direkten Rückmeldung, bei der ein 0-Bit eine gedrückte Taste bedeutet. Die Routine gibt die gleichen Tabellen aktualisiert zurück.

3.3 JUMP RESTORE

3.3.1 Die Aufgaben des Jump Restore Packs

Dieses Pack dient vor allem der Initialisierung der RAM-Vektoren im CPC. Sollten diese gezielt oder versehentlich verändert worden sein, so können sie mit dem Aufruf der Routine Jump Restore wiederhergestellt werden.

Die umfangreiche Adressentabelle, über die der Schneider-Computer verfügt, hat verschiedene Aufgaben.

1. Durch das Banking im CPC ist der Zugriff auf Betriebssystem-Routinen für Programme im RAM oder in einem oberen ROM nicht ganz einfach. Vor dem Ansprung muß gegebenenfalls noch das ROM eingeschaltet und nach Ausführung der Routine der alte Status wieder gesetzt werden. Dadurch, daß sämtliche Ansprünge des Betriebssystems über Restarts (siehe 3.1.6) erfolgen, können Benutzer-Programme nunmehr sehr einfach das Betriebssystem anspringen.

2. Wenn man größere Programme schreibt, die man z.B. vermarkten möchte, so stellt sich stets die Frage nach der Kompatibilität, sobald mehr als eine Version eines Computers auf dem Markt ist. Dies gilt beim Schneider-Computer natürlich ganz genauso, da dieser mittlerweile in drei Versionen (464, 664 und 6128) verkauft wird. Die Betriebssysteme der drei Maschinen sind zwar über weite Strecken logisch identisch, durch gewisse kleinere Änderungen kam es jedoch zu Verschiebungen der einzelnen Routinen. Würde man nun das Operating System direkt anspringen, so wäre ein Programm für den 464 auf dem 664 völlig unbrauchbar, weil die Ansprünge aufgrund der Verschiebungen natürlich falsch wären. Gleiches gilt für die (In-)Kompatibilität von CPC 664 und CPC 6128. Dem wird durch Sprungtabellen begegnet, in denen die einzelnen Ansprünge für die Routinen (man nennt diese auch "Vektoren") immer an der gleichen Stelle bleiben, jedoch selber durchaus auf verschiedene Einsprungadressen verweisen können.

3. Das Betriebssystem des CPC ist zwar ein recht gutes System insofern, als es eigentlich alle hardwaremäßig vorhandenen Möglichkeiten des Computers unterstützt. Dennoch kann es vorkommen, daß man anstelle einer Betriebssytem-Routine eine eigene Routine einbauen möchte, um z.B. neue Hardware zu unterstützen. Ein solcher Fall liegt z.B. beim Anschluß einer Floppy an den 464 vor. Da sich das Basic in einem ROM befindet, kann

man die Aufrufe der Ein-/Ausgaberoutinen nicht verändern. Würde das Basic das Betriebssystem direkt anspringen, so könnte man die Floppy von Basic aus nicht benutzen. Da das Basic jedoch nicht direkt, sondern über die Sprungtabelle die I/O-Routinen aufruft, kann das DOS anstelle der Cassettenroutinen die entsprechenden Ansprünge der Disketten-Routinen einsetzen. Die Sprungtabellen haben somit auch die Aufgabe, eine Erweiterung des Systems so zu ermöglichen, daß die bestehende Software die neuen Systemteile (die dann natürlich mit ein wenig Interface-Software ausgestattet sein müssen) ohne Änderung benutzen kann.

Dies alles bezieht sich auf die Haupttabelle der festen Betriebssystem-Ansprünge. Es gibt jedoch noch einige andere Sprungtabellen im CPC. Im Folgenden wollen wir alle Tabellen des Jump Restore Packs unter die Lupe nehmen.

3.3.2 Die Sprungtabellen im CPC

Wir unterscheiden vier verschiedene Tabellen im Schneider-Computer:

1. **Die Haupttabelle:** Hier liegen Vektoren für alle wesentlichen Routinen des Betriebssystems. Sie haben aus Gründen der Kompatibilität in jeder Version garantiert die gleiche Bedeutung. Die Haupttabelle geht im 464 von $BB00 bis $BD39, im 664 bis $BD5A und im 6128 bis $BD5D, wegen der zusätzlichen Routinen, die aber für alle künftigen Versionen konstant bleiben sollen. Die Ansprünge werden realisiert durch ein RST1 mit folgender Adresse und ROM-Switches. Die ROM-Switches sind normalerweise so eingestellt, daß das obere ROM ausgeschaltet ist. Dies ist z.B. für die Bildschirmroutinen von Bedeutung, da diese ja auf das Video-RAM zugreifen müssen. Das untere ROM ist dabei natürlich an. Die Vektoren der Haupttabelle können direkt vom Benutzer angesprungen werden, Dies ist, wie wir noch sehen werden, nicht unbedingt selbstverständlich.

2. **Die Nebentabelle:** Die Vektoren dieser Tabelle variieren in ihren Bedeutungen (zwischen 464 und 664/6128), d.h. Software, die auf allen drei Maschinen laufen soll, sollte die Benutzung dieser Segmente des Betriebssystems vermeiden. Durch die Nebentabelle können der Editor und die Arithmetik angesprungen werden. Im 464 also sowohl Fließkomma- als auch Integerarithmetik. Im 664/6128 ist die Integerarithmetik in das obere ROM übernommen worden. Die Nebentabelle hat also nur Bedeutung für Software, die speziell für das aktuelle System geschrieben wurde. Im RAM liegt

sie beim 464 von $BD3A bis $BDCC, im 664 von $BD5B bis $BDBD und im 6128 von $BD5E bis $BDC0. Die Ansprünge selber werden über ein RST5 realisiert, das den Status des oberen ROMs nicht verändert. Die Fließkomma-Arithmetik kann dies z.B. benutzen, indem ihr ein Zeiger in das obere ROM gegeben wird, um ihr für Berechnungen bestimmte Werte zu übergeben. Die einzige Ausnahme bildet Edit. Diese Routine wird - wie auch die Einträge in der Haupttabelle - mit einem RST1 angesprungen. Dies geschieht, weil die Betriebssystem-Routinen sich untereinander nicht über die Haupt- oder die Nebentabelle aufrufen, sondern sich direkt anspringen. Da der Editor auch Packs anspringt, die z.B. auf das Video-RAM zugreifen müssen, muß zu diesem Zweck das obere ROM für Edit ausgeschaltet sein. Dies wäre mit einem RST5 nicht zu garantieren. Wie auch schon die Vektoren der Hauttabelle, so können auch die der Nebentabelle direkt vom Benutzerprogramm angesprungen werden.

3. **Die Indirections:** Im Gegensatz zu den beiden ersten Tabellen sind die Indirections keine RST-Vektoren, sondern werden ganz einfach über JumPs realisiert. Dies setzt dann natürlich voraus, daß das untere ROM eingeschaltet ist, was die Indirections weitgehend unbrauchbar für den Aufruf durch ein Benutzer-Programm macht. Tatsächlich sind sie auch nicht dazu gedacht, von einem User-Programm angesprungen zu werden. Sie sollen lediglich von bestimmten Stellen des Betriebssystems aus aufgerufen werden. Trotz dieser Einschränkungen können sie von großem Wert für den Benutzer sein. Indirections für Routinen wurden nämlich immer dort eingeführt, wo irgendwelche grundlegenden Operationen ausgeführt werden, die man zu bestimmten Zwecken sinnvoll durch andere ersetzen könnte. Ein Beispiel dafür ist die Routine MC Wait Printer: Diese Routine gibt ein Zeichen an das Centronics-Interface aus, wobei sie eine Zeit lang wartet, falls die angeschlossene Einheit noch nicht fertig sein sollte. Will man jetzt aber die gesamte Drucker-Ausgabe beispielsweise auf ein selbstgebautes serielles Interface lenken, so schreibt man eine kurze Bearbeitungs-Routine. Man "verbiegt" die Indirection derart, daß sie auf die neue Routine zeigt. Die Indirections sind also für den User nur dadurch von Wert, daß er sie verändern kann, um so Änderungen in der I/O des CPC vorzunehmen.

4. **Die Kernel Hi Jumps:** Auch diese Tabelle ist nicht mit Restarts, sondern mit Jumps realisiert. Dies ist aber auch schon das einzige, was die Hi Jumps mit den Indirections verbindet. Denn zum einen

zeigen die Vektoren nicht auf I/O-Routinen, zum anderen sind sie für das Benutzerprogramm vor allem dadurch zu benutzen, daß sie angesprungen werden. Die Routinen nämlich, die durch die Hi Jumps zusammengefasst werden, sind Banking Routinen, die demnach auch nicht im unteren ROM, sondern in den zentralen 32 KByte zu finden sind. Logischerweise zeigen dann auch die Hi Jumps in diesen Bereich. Da daher auch nicht der Status der ROMs von Interesse ist, können die Hi Jumps von jeder Stelle im System ohne Vorbehalte angesprungen werden. Eine Veränderung der Hi Jumps scheint kaum sinnvoll, da die entsprechenden Routinen eigentlich nicht mehr optimiert oder sinnvoll verändert werden können. Diese Tabelle hat demnach vor allem die Aufgabe, eine Adressenunabhängigkeit der Kernel-Routinen herzustellen. Die Hi Jumps werden vom Kernel ins RAM kopiert. Das Jump Restore hat mit dieser Sprungtabelle eigentlich nichts zu tun. Wir haben sie lediglich der Systematik halber an dieser Stelle erwähnt.

3.3.3 Die Benutzung der Haupt- und Nebentabelle

Da die Haupttabelle vor allem dazu dient, Programme transportabel zu machen, d.h. sie derart zu gestalten, daß sie auf verschiedenen Versionen des Betriebssystems laufen, ist es beim Entwurf von Programmen nötig, einige Konventionen einzuhalten, die vom Hersteller vorgegeben wurden. Nur durch strenges Einhalten solcher Normen ist gewährleistet, daß die Benutzer-Software wirklich systemunabhängig ist. Die Konventionen sollten auch für die Nebentabelle eingehalten werden. Dies erhöht die Systematik und die "Sauberkeit" des Programms.

Eine wichtige Vereinbarung ist die, daß Aufrufe von Betriebssystem-Vektoren nur an dem dafür vorgesehenen Ort geschehen sollen, d.h. man sollte nicht einen Vektor an eine andere Stelle ins RAM kopieren und ihn dann dort anspringen.

Wenn man Vektoren umdefinieren möchte so sollte das so geschehen, daß man den laufenden Vektor in einen sicheren Bereich kopiert und dann den neuen an dessen Stelle schreibt. Braucht man den neu definierten Vektor nicht mehr, so setzt man den alten wieder an seine Stelle.

Neben diesen Formalien ist natürlich darauf zu achten, daß eigene Routinen, die man anstelle von Operating System-Routinen als Vektoren setzt, möglichst die gleiche Parametrisierung haben wie diese. Von der Ebene des Anwenderprogrammes her, sollten diese beiden Routinen nicht zu unterscheiden sein.

Schreibt man eigene User-Programme, so sollte man aus Gründen der Übertragbarkeit versuchen, nur Vektoren der Haupttabelle zu benutzen, da bei verschiedenen Versionen nur diese als konstant garantiert sind.

3.3.4 Struktur des unteren ROMs

An der Struktur der Haupt- und der Nebentabelle kann man sehr gut auch die Struktur der Routinen im unteren ROM ableiten. So erkennt man einen Hauptteil, in dem alle Packs liegen, außer der Arithmetik und dem Editor, die ihrerseits in dem Nebenteil zusammengefaßt sind. Das Firmware Manual bezieht sich ausschließlich auf den Hauptteil, obschon man unter Firmware eigentlich sämtliche fest installierte Software in einem System versteht.

Dies verleiht dem Hauptteil des unteren ROMs einen besonderen Status. Nach der Nomenklatur, die der Hersteller vorschlägt, wird dieser Teil deshalb unter der Bezeichnung "Betriebssystem" (Operating System, OS) zusammengefaßt. Der Teil der Fimware, der sich im oberen ROM befindet, bildet das "Basic". Der Nebenteil des unteren ROMs bleibt hingegen unbenannt. Man erkennt das "zwischen den Stühlen sitzen" dieses Teils z.B. auch daran, daß sich im 464 das Integer-Pack noch im unteren ROM befindet, im 664 und im 6128 aus Platzgründen kurzerhand ins obere ROM getan wurde.

Bei der Strukturierung dieses Buches stellte sich die Frage, ob dieser Firmware-Teil ohne Name nun dem Basic oder dem OS zuzuschreiben ist oder gar separat behandelt werden sollte. Wie Sie sehen, entschieden wir uns dafür, ihn zusammen mit dem Betriebssystem zu beschreiben. Dafür gibt es mehrere Gründe: zum einen spricht natürlich die speicherplatzmäßige Zuordnung (d.h. die Unterbringung dieses Teils im OS-ROM) zum Betriebsystem dafür. Sie steht in Verbindung mit der Einführung von speziellen Vektoren für diese Routinen, die auch vom OS verwaltet werden (zusammen mit der Haupttabelle vom Jump Restore Pack). Dann spricht auch noch die sehr saubere Programmierung dieser Systemteile und ihre hohe Modularität für eine Verbindung zum Operating System. Eine solch hohe Modularität ist im Basic-ROM kaum zu finden.

3.4 Das SCREEN PACK (SCR)

3.4.1 Allgemeines

Das Screen-Pack im CPC ist die unterste Ebene der Bildschirmverwaltung. In diesem Pack finden sämtliche Schreib- und Lesezugriffe auf den Bildschirmspeicher statt. Die Bildschirm-Adreßberechnung wird hier ebenso durchgeführt wie das soft- oder hardwaremäßige Scrollen des Bildschirms nach oben oder nach unten. Alle diese Funktionen werden vom Text- und Graphics-Pack benutzt, die die nächsthöhere Ebene der Bildschirmverwaltung darstellen.

Wie ähnliche Routinen in den meisten anderen Packs, wird auch die Routine SCR INTIALIZE bei einem Kaltstart aufgerufen. Sie ruft SCR RESET auf, initialisiert den Bildschirmspeicher-Start, schaltet Mode 1 ein und löscht den Bildschirm. Die Routine SCR RESET legt die Farbstiftzuordnungen auf Voreinstellungs-(Default-)Werte und kopiert drei Indirections vom ROM ins RAM.

3.4.2 Die Auswahl der verschiedenen Modes

Bevor eine Operation im Screen-Pack durchgeführt werden kann, muß erst einmal der Mode festgelegt werden, in dem die Darstellung erfolgen soll. Die Mode-Nummer wird, entsprechend des beim Basic-Befehl MODE übergebenen Werts, dem Screen-Pack über die Routine SCR SET MODE mitgeteilt. Diese Mode-Nummer löscht nach dem Einschalten des gewünschten Mode den Bildschirm mittels SCR MODE CLEAR, da der alte Bildschirmspeicherinhalt, im neuen Mode dargestellt, zu ungewöhnlichen Pixelmustern führen kann. Weiterhin wird innerhalb eines Bytes eine Tabelle von Bitmasken für die Pixelauswahl erstellt. In der ersten dieser Bitmasken sind alle die Bits gesetzt, die die Darstellung des ersten, ganz links stehenden, Pixels bestimmen. Bei Mode 0 sind dies vier Bits (Maske $AA), bei Mode 2 dagegen nur ein Bit (Maske $80). Es folgen die Bitmasken für die übrigen Pixels im Byte. Die Zahl der Bitmasken, also der Pixels pro Byte, beträgt je nach Mode zwei, vier oder acht.

Da die Codierung der Farben, die später noch beschrieben wird, vom ausgewählten Mode abhängt, werden die Farbmasken bei allen Text- und dem Graphik-Window decodiert und anschließend für den neuen Mode codiert. Die Text-Windows werden außerdem auf Default-Werte gesetzt, da die Window-Spaltengrenzen im neuen Mode eventuell nicht mehr zulässig sind.

3.4.3 Die Textzeichen-Matrizen

Die Informationen, wie ein Textzeichen auf dem Bildschirm aufgebaut ist, werden dem Screen-Pack vom Text-Pack in Form einer acht Byte langen Zeichenmatrix übergeben. Diese Matrix entspricht einer in Basic mit dem Befehl SYMBOL definierten Matrix. Die Matrixbytes beschreiben ein Feld von acht mal acht Pixels; ein gesetztes Bit in der Matrix bedeutet ein gesetztes Pixel auf dem Bildschirm. Da aber nur in Mode 2 ein Byte des Bildschirmspeichers acht Pixels darstellt, müssen in den anderen Modes die Matrixbytes noch aus der gepackten Form (ein Bit pro Pixel) in eine ungepackte Form gebracht werden. Mit Hilfe der oben beschriebenen Bitmasken werden für jedes gesetzte Pixel zwei Bits (bei Mode 1) oder vier Bits (bei Mode 0) in der ungepackten Matrix gesetzt. Bei Mode 2 hingegen braucht die gepackte Matrix nur kopiert zu werden.

Die ungepackte Matrix wird an die Routine zurückgegeben, die das Screen-Pack aufruft, in der Regel an das Text-Pack. Sie kann nach Modifikation entsprechend der ausgewählten Farben direkt in den Bildschirmspeicher übertragen werden. Das Konvertieren einer Matrix in seine ungepackte Form geschieht mit Hilfe von SCR UNPACK und das Packen einer direkt aus dem Bildschirmspeicher gelesenen Matrix mit SCR REPACK.

3.4.4 Die Verwaltung der Farben

Wie schon in Kapitel 1.3 beschrieben, geht die Erzeugung der Farben im CPC auf mehreren Ebenen vonstatten. Unter den vom Gate Array erzeugten 32 Farben sind fünf Farben, die nicht als eigenständige neue Farben bezeichnet werden können. Sie weisen keinen erkennbaren Unterschied zu bestimmten, aus den 27 restlichen Farben auf. Die Farbwerte dieser Farben (1, 8, 9, 16, 17) sind für den Benutzer jedoch nicht so leicht zu merken. Daher wird mit einer Tabelle aus der vom Benutzer übergebenen Farbnummer erst der ans Gate Array übergebene Farbwert berechnet. Die fünf erwähnten Farben haben die Nummern 27 bis 31.

Je nach Mode sind 2, 4 oder 16 verschiedene Farben gleichzeitig darstellbar (d.h. 2, 4 oder 16 "Farbstifte" verwendbar). Die Zuordnung von Farbnummern zu den verschiedenen Farbstiften geschieht in Basic mit dem INK-Befehl und im Screen-Pack mit der Routine SCR SET INK. Diese Routine wandelt die beiden übergebenen Farbnummern in Farbwerte um. Die Farbwerte werden in zwei Tabellen abgelegt. Aus einer dieser Tabellen werden die Farbwerte dann ans Gate Array übergeben. Entsprechend erfolgt die Festlegung der Rahmenfarbe(n) (Basic-Befehl BORDER) mit SCR SET BORDER.

Da der CPC die Möglichkeit bietet, jedem Farbstift zwei Farben zuzuordnen, die sich ständig abwechseln (blinken), können die Farbwerte nicht direkt ans Gate Array übergeben werden. Die Übergabe findet in einer Event-Routine statt, die die Werte aus einer der zwei Farbwerttabellen entnimmt. Der zugehörige Event-Block ist in der Frame-Fly-Chain eingehängt. Somit werden die Farben nur nach einem Strahlrücklauf gewechselt. Die Zahl der Aufrufe der Event-Routine, die bis zum nächsten Farbwechsel vergehen soll, wird mit SCR SET FLASHING festgelegt. Ein niedriger Wert bedeutet hier einen raschen Farb(tabellen)wechsel.

Wenn ein Pixel auf dem Bildschirm in der Farbe eines bestimmten Farbstiftes dardestellt werden soll, so muß die Nummer des Farbstifts (codiert) an die entsprechende Stelle im Bildschirmspeicher geschrieben werden. Mit der Routine SCR INK ENCODE kann eine Farbstiftnummer in die entsprechende Farbmaske gewandelt werden. In dieser Farbmaske ist die Farbstift-Nummer je nach Mode zwei, vier oder achtmal jeweils an den Bit-Positionen für ein Pixel enthalten. Die Farbmasken für die Pen- und Paper-Farbstifte ergeben, verknüpft mit der Maske für die Pixelauswahl, dann das in den Bildschirmspeicher zu schreibende Byte. Die Pen- und Paper-Farbmasken werden jedoch nicht vom Screen-Pack, sondern vom Text- und Graphics-Pack verwaltet. Das Screen-Pack unterstützt nur die Umcodierung. Mit der Routine SCR INK DECODE kann aus der Maske wieder die Farbstift-Nummer berechnet werden, während SCR READ die Farbe eines bestimmten Pixels auf dem Bildschirm decodiert.

3.4.5 Die Adreßberechnung

Wie schon im Kapitel 1.2 beschrieben, besteht die Startadresse des Bildschirms aus zwei Teilen. Mit SCR SET BASE kann der 16-KByte-Block ausgewählt werden, in dem der Bildschirmspeicher liegen soll. Will man den durch SCR INITIALIZE eingestellten Wert $C000 für eigene Anwendungen ändern, so wäre aufgrund der Speicherbelegung des CPC nur der Wert $4000 sinnvoll. Die Routine für das Hardware-Scrolling verschiebt durch SCR SET OFFSET innerhalb des durch SCR BASE ausgewählten Bereichs die Bildschirm-Startadresse. SCR OFFSET kann jedoch wegen der Hardware des CPC nur innerhalb der ersten 2 KByte des ausgewählten 16-KByte-Blocks und nur in 2-Byte-Schritten gesetzt werden.

Im CPC 664/6128 ist es mit Hilfe der Routine SCR SET POSITION möglich, neben dem sichtbaren Bildschirminhalt weitere, unsichtbare Bildschirme aufzubauen. Dies kann z.B. nützlich sein, wenn man schnell zwischen verschiedenen Bildschirmen umschalten will oder die Bildschirm-Aufbauphase unsichtbar bleiben soll. An SCR SET POSITION werden die Werte für SCR OFFSET und SCR BASE übergeben. SCR SET POSITION

speichert diese Werte zwar in den entsprechenden Systemvariablen ab, so daß die Aktionen des Screen- und damit auch des Text- und Graphics-Packs mit den neuen Werten durchgeführt werden. Der Hardware (also dem Gate Array) werden SCR OFFSET und SCR BASE jedoch nicht neu übergeben, so daß nach SCR SET POSITION der gleiche Bildschirm sichtbar bleibt.

Die Bildschirmadresse für ein, durch Graphik-Koordinaten angegebenes, Pixel wird von der Routine SCR DOT POSITION berechnet. Diese Routine liefert auch noch die Maske für die Auswahl des gewünschten Pixels innerhalb des Bildschirmbytes. Für Textpositionen verwendet man dagegen die Routine SCR CHAR POSITION. Sie errechnet die Adresse des linken oberen Pixels eines Zeichens aus Zeile und Spalte und gibt keine Pixelauswahl- Maske zurück, da bei Ausgabe eines Textzeichens alle Pixels im Byte verändert werden und die Masken deshalb einer ungepackten Zeichen-Matrix entnommen werden. Die an diese und andere Routinen übergebenen Text-Koordinaten müssen innerhalb bestimmter Grenzen liegen, die vom Mode abhängen. Diese Grenzkoordinaten kann man durch die Routine SCR CHAR LIMITS erfahren.

Bei der Ausgabe eines Textzeichens müssen in Mode 0 und Mode 1 mehrere Bytes in einer Rasterzeile verändert werden. Auf jeden Fall aber werden Pixels in acht verschiedenen Rasterzeilen verändert, da ein Zeichen mit acht mal acht Pixeln dargestellt wird. Da die Adresse des nächsten Bildschirmbytes wegen des möglichen Übertrags zu den RA-Bits (Bits 10 bis 12, siehe Kapitel 1.2) nicht immer die folgende Adresse ist, kann mit Hilfe der Routinen SCR NEXT BYTE und SCR PREV BYTE die Adresse des nächsten bzw. vorigen zu bearbeitenden Bytes berechnet werden. Will man die Adresse der Position einer Rasterzeile über der jetzigen Position erfahren, so ruft man SCR PREV LINE auf, für die nächste Rasterzeile dagegen SCR NEXT LINE. Diese Routinen sind nicht nur bei Ausgabe eines Textzeichens sinnvoll einsetzbar. Auch beim Zeichnen einer senkrechten Pixel-Linie beispielsweise spart es Zeit, SCR NEXT LINE anstatt von SCR DOT POSITION für jeden Pixel neu aufzurufen.

3.4.6 Das Setzen der Pixels auf dem Bildschirm

Das Setzen von einem Pixel geschieht mit der Routine SCR WRITE. An sie werden die Bildschirmadresse, die Farbmaske der Pixelfarbe und die Maske für Pixelauswahl übergeben. Über letztere können übrigens auch mehrere Pixels gleichzeitig gesetzt werden. Es wird jedoch immer nur ein Byte bearbeitet. Mit der Routine SCR ACCESS kann festgelegt werden, ob bei SCR WRITE die Pixels direkt gesetzt werden oder ob eine AND-, OR- oder XOR-Verknüpfung mit dem bestehenden Bildschirminhalt durchge-

führt werden soll. Will man die Pixels auf jeden Fall direkt - ohne Rücksicht auf den vorherigen Bildschirminhalt - setzen, so sollte man anstatt SCR WRITE die Routine SCR PIXELS benutzen.

Komplexere Aufgaben bewältigen die Routinen SCR FILL BOX und SCR FLOOD BOX. Sie füllen einen rechteckigen Ausschnitt des Bildschirms mit einer bestimmten Farbe. Während SCR FILL BOX neben der Farbmaske Textzeilen- und Spalten zur Begrenzung des Rechtecks übergeben werden müssen, sind bei SCR FLOOD BOX die Bildschirmadresse, Zahl der Rasterzeilen und Zahl der Bytes pro Rasterzeile gefragt. Mit SCR CHAR INVERT kann durch Vertauschen zweier Pixelfarben ein Textzeichen invertiert werden, was zur Cursordarstellung genutzt wird. Schließlich werden von SCR HORIZONTAL und SCR VERTICAL waagerechte bzw. senkrechte Linien gezeichnet. Diesen beiden Routinen müssen Graphik-Koordinaten übergeben werden.

3.4.7 Das Scrolling

Soll der gesamte Bildschirm nach oben oder unten gescrollt werden, so benutzt man dafür die Routine SCR HARDWARE ROLL. Sie ist schneller als die Routine SCR SOFTWARE ROLL, mit der man ein nicht den ganzen Bildschirm umfassendes Window scrollen kann. Die Windowgrenzen müssen hierzu an die Routine übergeben werden.

3.5 Das TEXT SCREEN PACK (TXT)

3.5.1 Allgemeines

Dieses Pack hat die Aufgabe, die acht verschiedenen Windows zu verwalten und eine Zeichenausgabe und Cursordarstellung auf dem Bildschirm zu ermöglichen. Es benutzt zur Ausführung dieser Aufgaben Routinen aus dem Screen-Pack.

Wie in jedem Pack, so finden sich auch hier die Routinen TXT INITIALIZE und TXT RESET. Während TXT INITIALIZE die Parameter für sämtliche Windows initialisiert, kopiert TXT RESET einige Indirections und initialisiert die Steuerzeichen-Sprungtabelle.

3.5.2 Die Verwaltung der Windows

Das Text-Pack erlaubt die Definition von acht Bildschirmfenstern (Windows). Die Länge eines Parameter-Blocks eines Windows beträgt im CPC 464 15 Byte. Der Aufbau ist wie folgt:

00	Cursorzeile (absolut)
01	Cursorspalte (absolut)
02	0 = Hardware-Scrolling, sonst Software-Scrolling
03	obere Windowgrenze
04	linke Windowgrenze
05	untere Windowgrenze
06	rechte Windowgrenze
07	Scrolling-Zähler
08	Cursor-Flag
	b0: 0 = enabled, 1 = disabled
	b1: 0 = ON, 1 = OFF
09	VDU-Flag (0 = keine Zeichenausgabe)
0A	Pen-Farbmaske
0B	Paper-Farbmaske
0C/0D	Indirection für Hintergrundmodus
0E	0 = Ausgabe auf Textcursorposition (TAGOFF)
	sonst Ausgabe auf Graphikcursorposition (TAG)

Im CPC 664 und im CPC 6128 ist ein Window-Parameter-Block nur 14 Byte lang:

00	Cursorzeile (absolut)
01	Cursorspalte (absolut)
02	0 = Hardware-Scrolling, sonst Software-Scrolling
03	obere Windowgrenze
04	linke Windowgrenze
05	untere Windowgrenze
06	rechte Windowgrenze
07	Scrolling-Zähler
08	Cursor-/VDU-Flag
	b0: 0 = enabled, 1 = disabled b1: 0 = ON, 1 = OFF
	b7: 0 = VDU enabled, 1 = VDU disabled

09	Pen-Farbmaske
0A	Paper-Farbmaske
0B/0C	Indirection für Hintergrundmodus
0D	0 = Ausgabe auf Textcursorposition (TAGOFF) sonst Ausgabe auf Graphikcursorposition (TAG)

Die Bedeutung einiger Parameter wird später bei den Routinen, die sie beeinflussen, erklärt. Wichtig ist zunächst die Verwaltung der verrschiedenen Parameter-Blöcke. Jedes der acht Windows hat einen 14 bzw. 15 Byte langen Parameter-Block. Zusätzlich existiert ein eigener Parameter-Block für das aktuell ausgewählte Window. Das hat den Vorteil, daß die Parameter des aktuellen Windows einfach geändert werden können, da sie auf festen Adressen liegen. Fast alle Änderungen der Parameter und alle Bildschirm-Aktionen durch die Routinen des Text-Packs beziehen sich auf das aktuelle Window. Auf die Ausnahmen hiervon wird im folgenden hingewiesen.

Will man die Parameter eines Windows ändern oder etwas über das Window ausgeben, so muß man das Window mit TXT STR SELECT als aktuelles Window setzen. Der Parameter-Block des Windows wird in den aktuellen Parameter-Block kopiert. Der aktuelle Parameter-Block wird vorher in den Parameter-Block des zuvor ausgewählten Windows zurückkopiert, damit die gemachten Änderungen auch bestehen bleiben. Wenn beispielsweise nur die Pen-Farbe des zweiten Windows geändert werden soll, so wählt man mit TXT STR SELECT das zweite als aktuelles Window aus. TXT STR SELECT gibt die Nummer des vorher ausgewählten Windows zurück. Man ändert nun im aktuellen Window die Pen-Farbe mittels TXT SET PEN und wählt wieder das vorher ausgewählte Window aus. Die Änderung des Pen-Wertes wird dann in den Parameter-Block des zweiten Windows kopiert.

Mit der Routine TXT SWAP STREAMS können die Parameter-Blöcke zweier Windows direkt vertauscht werden.

3.5.3 Die Window-Grenzen

Die Grenzen (links, rechts, oben und unten) des aktuellen Windows können mit TXT WIN ENABLE neu gesetzt werden. Die an diese Routine übergebenen Zeilen- und Spaltenwerte sind absolute Werte. Die linke obere Ecke des Bildschirms ist Position 0/0. Die Zeilengrenzen werden von der Routine in den Bereich 0..24 und die Spaltengrenzen, je nach Mode, in den Bereich 0..19, 0..39 oder 0..79 forciert. Die meisten Koordinaten, die sich auf das Window beziehen, werden relativ zum Window angegeben. Die linke obere Ecke des Windows ist dann die Position 1/1. Mit TXT GET WINDOW können die Grenzen des aktuellen Windows abgefragt werden. TXT VALIDATE forciert eine Bildschirmposition in die Windowgrenzen.

3.5.4 Die Cursorsteuerung

Da auf acht verschiedenen Windows Zeichen ausgegeben werden können, existieren auch acht verschiedene Cursorpositionen. Die Cursorposition des aktuellen Windows kann mit drei Routinen direkt gesetzt werden: TXT SET ROW setzt die Zeile, TXT SET COLUMN die Spalte und TXT SET CURSOR Zeile und Spalte neu. Zeile und Spalte werden relativ zum Window angegeben. Mit TXT GET CURSOR kann die Cusorposition abgefragt werden.

Das Screen-Pack kann mit TXT DRAW CURSOR den Cursor an der aktuellen Position sichtbar machen und mit TXT UNDRAW CURSOR verschwindet er wieder. Von der Benutzerebene aus geschieht dies durch die Routinen TXT PLACE CURSOR und TXT REMOVE CURSOR. Die ersteren Routinen werden benutzt, um bei der Zeichenausgabe den Cursor mit den Zeichen weiterlaufen zu lassen. Wenn man die Indirections für TXT DRAW CURSOR bzw. TXT UNDRAW CURSOR nicht verändert hat, so benutzen beide die gleiche Routine, die den Cursor invertiert. Ebenso springen die Betriebssystemvektoren für TXT PLACE CURSOR und TXT REMOVE CURSOR die gleiche Routine an. Deshalb darf keine dieser Routinen zweimal hintereinander aufgerufen werden, wenn die ursprüngliche Funktion gewahrt werden soll.

Der Cursor ist von höherer Ebene auch mit TXT CUR ON und TXT CUR OFF ein- und ausschaltbar. Hier ist es möglich, die gleiche Routine zweimal hintereinander aufzurufen, der zweite Aufruf bewirkt dann nichts. Nach TXT CUR OFF sind TXT DRAW CURSOR und TXT UNDRAW CURSOR bis zum nächsten TXT CUR ON wirkungslos. TXT CUR ON und TXT CUR OFF werden vom Basic beim Warten auf eine zweite Taste nach der ESC-Taste und vom Editor aufgerufen.

Ebenso funktionieren die Routinen TXT CUR ENABLE und TXT CUR DISABLE. Über sie wird der Cursor ausgeschaltet, wenn die gesamte Bildschirmausgabe durch TXT VDU DISABLE ausgeschaltet ist. Der Cursor ist also nur sichtbar, wenn sowohl TXT CUR ENABLE als auch TXT CUR ON aufgerufen wurden. Nach TXT INITIALIZE ist in allen Windows der Cursor OFF und ENABLED.

3.5.5 Die Verwaltung der Zeichenmatrizen

Zur Darstellung von Zeichen auf dem Bildschirm werden Zeichenmatrizen benötigt. Sie geben an, welche Pixels in dem acht mal acht Pixels großen Feld, das ein Zeichen darstellt, gesetzt werden müssen. Da die Information für ein Pixel ein Bit belegt, ist eine Zeichenmatrix 64 Bit = 8 Byte groß. Jedes Byte aus der Matrix stellt eine Rasterzeile des Zeichens und das

höchste Bit innerhalb eines Bytes stellt das ganz links stehende Pixel in der Zeile dar.

Die Zeichenmatrizen sind nicht window-spezifisch. Bei allen Windows werden gleiche Zeichen auch durch gleiche Matrizen dargestellt. Nach Ausführung von TXT INITIALIZE werden die Zeichenmatrizen im ROM benutzt (von $3800 bis $3FFF). Es besteht jedoch die Möglichkeit, sich eigene (User-)Matrizen im RAM zu definieren. Dazu muß man der Routine TXT SET M TABLE übergeben, ab welchem Zeichen die Matrizen umdefiniert werden sollen (analog dem Basic-Befehl SYMBOL AFTER) und ab welcher Adresse die neu definierten Matrizen im RAM abgelegt werden sollen. Mit TXT SET MATRIX kann danach die Matrix für ein Zeichen (analog dem Basic-Befehl SYMBOL) neu gesetzt werden. TXT GET M TABLE gibt ein Flag, ob User-Matrizen definiert wurden und gibt die Parameter der User-Matrizen zurück. TXT GET MATRIX schließlich stellt für ein bestimmtes Zeichen fest, ob eine User-Matrix oder die ROM-Matrix benutzt werden soll und berechnet die Adresse der zugehörigen Matrix, um das Zeichen dann z.B. auf dem Bildschirm ausgeben zu können.

3.5.6 Die Farben

Die Pen- und Paper-Farbstift-Nummern werden durch die Routinen TXT SET PEN und TXT SET PAPER gesetzt und durch TXT GET PEN und TXT GET PAPER wieder abgefragt. Gespeichert werden jedoch aus Gründen der Zeitersparnis nicht die Farbstift-Nummern, sondern die codierten Farbmasken. Mit TXT INVERSE werden Pen- und Paper-Farbstift vertauscht.

Durch TXT SET BACK kann man den Hintergrund-Modus wählen. Die Pixels, deren zugehöriges Bit der Zeichenmatrix gesetzt ist, werden auf jeden Fall in der Pen-Farbe gesetzt. Vom Hintergrund-Modus hängt nun die Darstellung der Pixels ab, deren Bit in der Matrix gelöscht ist. Wenn der Hintergrund-Modus transparent ist, dann werden diese Pixels nicht verändert, andernfalls werden sie in der Paper-Farbe gesetzt. Der letztere Fall ist der Default. Den gerade ausgewählten Modus kann man durch TXT GET BACK erfahren.

3.5.7 Die Ausgabe von Zeichen auf den Bildschirm

Mit der Routine TXT WRITE CHAR wird ein Zeichen an einer bestimmten absoluten Bildschirmposition ausgegeben. Die Codes von $00 bis $1F (normalerweise Steuerzeichen) werden mit eigenen Symbolen dargestellt.

TXT WR CHAR gibt ein Zeichen an der Cursorposition des aktuellen Windows mittels TXT WRITE CHAR aus, setzt die Cursorposition für das

nächste Zeichen und invertiert den Cursor vorher und hinterher, damit er eine Position weiterrückt. Mit TXT VDU DISABLE kann die Zeichenausgabe unterbunden werden, TXT WR CHAR ist dann wirkungslos.

TXT OUTPUT springt zur Indirection TXT OUT ACTION. TXT OUT ACTION gibt ein Zeichen an der Cursorposition des aktuellen Windows mittels TXT WR CHAR aus. Steuerzeichen werden jedoch durch eigene Routinen ausgeführt und nicht direkt dargestellt. Die Ausgabe von Steuerzeichen läßt sich im CPC 464 bei TXT OUT ACTION nicht durch TXT VDU DISABLE unterbinden. Im CPC 664/6128 werden dagegen alle Steuerzeichen bis auf ESC und das Zeichen für TXT VDU ENABLE nach einem TXT VDU DISABLE unterdrückt. Die Auswertung der Steuerzeichen erfolgt über einen speziellen Buffer, in dem die für das Steuerzeichen notwendigen weiteren Zeichen gesammelt werden. Die Ausführung eines Steuerzeichens geschieht mit Hilfe einer Tabelle im RAM, die für jedes Steuerzeichen die Zahl der zusätzlich benötigten Zeichen, die Ausführadresse und (nur beim CPC 664/6128) trotz TXT VDU DISABLE ein Flag für die Ausgabe enthält. Man kann sich also seine eigenen Steuerzeichen definieren, wenn man diese Tabelle ändert. Die Adresse der Tabelle ist von der Routine TXT GET CONTROLS zu erfahren.

Zeichen werden von TXT OUTPUT und TXT OUT ACTION normalerweise an der Text-Cursorposition ausgegeben. Es ist jedoch auch möglich, Zeichen an der Graphik-Cursorposition auszugeben. Dazu benutzt TXT OUT ACTION die Routine GRA WR CHAR des Graphic-Packs. Steuerzeichen werden dann allerdings, wie bei TXT WRITE CHAR, mit eigenen Symbolen dargestellt. Für das aktuelle Window kann man diese Darstellungsart mit der Routine TXT SET GRAPHIC ein- und ausschalten, die auch von den Basic-Befehlen TAG und TAGOFF aufgerufen wird.

3.5.8 Das Lesen von Zeichen auf dem Bildschirm

Beim Editieren von Programmen mit dem Copy-Cursor des Editors muß beim Drücken der Copy-Taste ein Zeichen vom Bildschirm übernommen werden. Die Routine TXT UNWRITE liest ein Zeichen an einer beliebigen absoluten Position, indem die aus dem Bildschirm gelesene Pixel-Matrix mit allen vorhandenen Zeichen-Matrizen verglichen wird. TXT RD CHAR liest ein Zeichen an der Cursorposition des aktuellen Windows und ruft dazu TXT UNWRITE über eine Indirection auf.

3.6 Das GRAPHICS SCREEN PACK (GRA)

3.6.1 Allgemeines

Dieses Pack stellt dem Benutzer einige Routinen zum Zeichnen von Linien und Punkten auf dem Bildschirm zur Verfügung. Es benutzt zur Ausführung seiner Aufgaben Routinen aus dem Screen-Pack.

Die Routine GRA INITIALIZE setzt den Pen- und Paper-Wert neu und initialisiert das Graphik-Window, während GRA RESET einige Indirections ins RAM kopiert.

3.6.2 Das Graphik-Window

Die Grenzen des Graphik-Windows können mit GRA WIN WIDTH (für die X-Koordinaten, linke und rechte Grenze) und GRA WIN HEIGHT (für die Y-Koordinaten, obere und untere Grenze) gesetzt werden. Die X-Koordinaten werden in den Bereich 0..639 forciert, die Y-Koordinaten in den Bereich 0..399. Die Einteilung in 640 mal 400 Punkte ist in allen drei Modes gleich, obwohl eine Auflösung von 640 mal 400 Pixels nicht erreicht wird. Die Position 0/0 ist bei diesen Routinen die linke untere Ecke des Bildschirms. Mit GRA GET WINDOW WIDTH und GRA GET WINDOW HEIGHT können die Grenzen wieder abgefragt werden. Abgespeichert werden die Grenzen jedoch als reale Koordinaten, d.h. die Koordinaten entsprechen auch der tatsächlichen Auflösung.

Mit der Routine GRA SET ORIGIN kann der Ursprung (Origin) des Koordinatensystems neu festgelegt werden. Alle nicht zur Origin- und Window-Definition benutzten Koordinaten werden als Koordinaten relativ zum gesetzten Origin interpretiert. Es können so je nach Window und Origin auch negative Koordinaten möglich sein.

Vor Ausführung der meisten Routinen werden die übergebenen Koordinaten in reale Koordinaten umgerechnet, indem zuerst die Origin-Koordinaten addiert und dann je nach Mode X- bzw. Y-Koordinaten durch 1, 2 oder 4 geteilt werden. Mit den entstandenen Koordinaten läßt sich leichter und schneller arbeiten. Im CPC 664/6128 kann auch der Benutzer diese Umrechnung mittels der Routine GRA FROM USER direkt durchführen lassen.

3.6.3 Das Zeichnen von Linien, Punkten und Zeichen

Ähnlich der acht Text-Cursor gibt es einen Graphik-Cursor. Die Cursorposition kann mit GRA MOVE ABSOLUTE gesetzt und mit GRA ASK CURSOR abgefragt werden. Mit GRA MOVE RELATIVE kann der Cursor um einen bestimmten Offset versetzt werden. Der Cursor wird also relativ zu sich selbst bewegt. Die Angabe der Koordinaten relativ zum Cursor geschieht unabhängig von der Tatsache, daß die Cursorposition selbst schon relativ zum Origin angegeben ist.

Ein einzelner Punkt kann mit GRA PLOT ABSOLUTE oder, bei cursorrelativen Koordinaten, mit GRA PLOT RELATIVE gesetzt werden. Diese beiden Routinen rufen eine Indirection zu GRA PLOT auf. GRA PLOT ABSOLUTE ist daher genau das gleiche wie GRA PLOT - wenn die Indirection nicht geändert wurde. Bei GRA PLOT RELATIVE werden dagegen vor dem Aufruf von GRA PLOT die Koordinaten umgerechnet.

Analog funktionieren GRA TEST ABSOLUTE, GRA TEST RELATIVE und GRA TEST sowie GRA LINE ABSOLUTE, GRA LINE RELATIVE und GRA LINE. Die Test-Routinen geben die Farbstiftnummer des Pixels an der übergebenen Position zurück, während die Line-Routinen eine Linie von der Cursorposition bis zur übergebenen Position ziehen. Die Cursorposition wird nach diesen Routinen jeweils neu gesetzt, entsprechend der übergebenen (End-)Koordinaten.

Während im CPC 464 GRA LINE die Linien immer durchgehend zieht, kann im CPC 664/6128 mit der Routine GRA SET LINE MASK ein Linienmuster bestimmt werden, so daß gestrichelte oder gepunktete Linien möglich sind. Bit 7 der an GRA SET LINE MASK übergebenen Linienmaske bestimmt, ob das 1., 9., 17. usw. Pixel der Linie in der Pen-Farbe gesetzt werden soll oder nicht. Bit 6 gilt für das 2., 10., 18. Pixel usw.

Wenn man im XOR-Mode für SCR WRITE arbeitet, die Linien also invertierend gezeichnet werden, kann es z.B. zum Zeichnen eines rechteckigen Rahmens nützlich sein, den ersten Punkt einer Linie nicht zu zeichnen, damit kein Punkt zweimal invertiert wird. Dazu muß man GRA SET FIRST ein entsprechendes Flag übergeben. Auch diese Möglichkeit ist nur im CPC 664/6128 implementiert.

3.6.4 Die Farben

Die Pen- und Paper-Farbstifte können mit den Routinen GRA SET PEN, GRA SET PAPER, GRA GET PEN und GRA GET PAPER gesetzt bzw. abgefragt werden. Die Routine GRA CLEAR WINDOW füllt das Graphik-Window mit der aktuellen Paper-Farbe.

Ähnlich wie im Text-Pack kann auch im Graphics-Pack der Hintergrund-Modus durch GRA SET BACK festgelegt werden, allerdings hier nur beim CPC 664/6128. Die Pixels, denen entsprechend der Linienmaske oder der Zeichenmatrix ein gesetztes Bit zugeordnet ist, werden auf jeden Fall in der Pen-Farbe gesetzt. Vom Hintergrund-Modus hängt nun die Darstellung der Pixels ab, deren Bit in der Maske bzw. Matrix gelöscht ist. Wenn der Hintergrund-Modus transparent ist, dann werden diese Pixels nicht verändert, andernfalls werden sie in der Paper-Farbe gesetzt. Der letztere Fall ist der Default.

3.6.5 Die Ausgabe eines Zeichens

GRA WR CHAR ist eine Routine, die von TXT OUT ACTION aufgerufen wird, wenn das TAG-Flag gesetzt war. Diese Routine gibt ein Zeichen an der Graphik-Cursorposition aus. Damit ist die Ausgabe von Zeichen an Positionen möglich, die keine Textpositionen sind. Nach der Ausgabe eines Zeichens wird die X-Koordinate der Graphik-Cursorposition um acht erhöht, so daß das nächste Zeichen rechts neben dem aktuellen Zeichen ausgegeben wird. Anstelle von Steuerzeichen stellt GRA WR CHAR die speziellen Zeichen für den Bereich von $00 bis $1F dar.

3.6.6 Das Ausfüllen einer Fläche

Die Routine GRA FILL im CPC 664/6128 füllt eine Fläche ausgehend von der Cursorposition mit einem übergebenen Farbstift. Als Grenzen der Fläche gelten dabei sowohl Pixels in der Farbe des übergebenen Farbstifts, als auch Pixels in der Pen-Farbe. Die Routine benötigt einen Buffer zum Abspeichern der Parameter von Teilen der Fläche (von einzelnen horizontalen Linien), die nicht sofort bearbeitet werden können. Adresse und Länge des Buffers müssen an GRA FILL übergeben werden.

Beim Füllen der Fläche wird in drei Fällen ein Fehler angezeigt (CY=0). Die Cursorposition kann außerhalb des Windows liegen oder das Pixel an der Cursorposition in einer Sperrfarbe gesetzt sein. Sperrfarben sind die Füll- und die Pen-Farbe, die ja die Fläche begrenzen. Weiterhin tritt ein Fehler auf, wenn der Buffer zu klein ist. Je komplexer die Fläche aufgebaut ist, desto mehr Buffer-Raum wird benötigt.

3.7 Der KEYBOARD MANAGER (KM)

3.7.1 Allgemeines

Das Keyboard Manager Pack verwaltet, wie auch schon aus dem Namen hervorgeht, die Eingabe von der Tastatur und erledigt alle Aufgaben, die damit verbunden sind. Dazu gehört, neben der Abfrage der Tastatur und der Berechnung der ASCII-Codes der gedrückten Tasten, auch die Verwaltung von Expansion Strings, des Put Back Buffers, der Tastenwiederholung (Key-Repeat) und der ASCII-Code-Tabellen für die Tastatur.

3.7.2 Die Erzeugung von Zeichen

Ein Programm, wie z.B. der Basic-Interpreter, benutzt zur Erzeugung von Zeichen den Keyboard-Manager. Dazu springt es eine Routine an (KM READ CHAR oder, wenn die Routine solange warten soll, bis ein gültiges Zeichen vorhanden ist, KM WAIT CHAR), die das aktuelle, im Keyboard Manager vorliegende Zeichen über den Akkumulator an das Programm zurückgibt. Dieses Zeichen nun kann aus drei verschiedenen Quellen stammen:

1. **Der Put Back Buffer:** Der Put Back Buffer ist ein Zwischenspeicher, der jeweils nur ein Zeichen enthält. Er dient dem übergeordneten Programm dazu, ein Zeichen vorläufig zurückzustellen, das ihm vom Keyboard Manager geliefert wurde und das es noch einmal braucht. Wenn das Programm dann das nächste Mal nach einem Zeichen fragt, wird es dieses zurückgestellte Zeichen wieder geliefert bekommen. Die Routine zum Zurückstellen eines Zeichens heißt KM CHARacter RETURN.

2. **Die Expansion Strings:** Expansion Strings sind Zeichenketten, die jeweils einem ASCII-Code von $80-$9F zugeordnet sind, und die Zeichen für Zeichen nacheinander ausgegeben werden, wenn der Keyboard Manager auf einen solchen Code stößt. Es existiert im Schneider ein bestimmter Speicherbereich im RAM, der "Expansion String Buffer", in dem alle 31 Expansion Strings in der richtigen Reihenfolge eingetragen sind. Wenn nun ein Expansion String aktiv ist (der Keyboard Manager stellt das dadurch fest, daß an einer bestimmten Stelle ein gültiger Expansion String-Code steht), so wird das laufende Zeichen (dessen Nummer im String ebenfalls an einer Stelle im System-RAM steht) mit der Routine KM GET EXPAND geholt, die Nummer des laufenden Zeichens erhöht und das Zeichen aus dem Expansion String an das aufrufende Programm zurückgegeben. So wird, wenn einmal ein Expansion String aktiviert

wurde, der String nach und nach, bei jedem Aufruf von KM READ CHAR ein Zeichen, ausgegeben, ohne daß eine Taste von der Tastatur "dazwischen kommen" kann.

3. **Die Tastatur:** Wenn weder ein Zeichen im Put Back Buffer zu finden ist, noch ein Expansion String aktiviert ist, wird das nächste Zeichen mit KM READ KEY von der Tastatur gelesen. Wenn der von KM READ KEY gelieferte ASCII-Code ein gültiger Expansion String Code ist (also von $80-$9F geht), dann wird der entsprechende Expansion String aktiviert. Dazu wird der Code zusammen mit der Zeichennummer Null (für das erste Zeichen im Expansion String) gesetzt. Daraufhin beginnt KM READ CHAR von neuem. Ist das Zeichen von der Tastatur jedoch kein Expansion Code, so wird es einfach an das aufrufende Programm zurückgegeben.

Ist selbst auf der Tastatur kein Zeichen zu finden, gibt KM READ CHAR ein gelöschtes Carry als Kennzeichen zurück. Gleiches macht KM READ KEY, wenn keine Taste gedrückt ist. Will der Benutzer solange warten, bis ein Zeichen da ist bzw. eine Taste gedrückt wurde, so kann er KM WAIT CHAR bzw. KM WAIT KEY anspringen.

Betrachten wir jetzt jedoch einmal die Tastaturabfrage ein wenig genauer. Wie schon erwähnt wird, um eine Taste von der Tastatur zu lesen, KM READ KEY angesprungen. Diese Routine, so wurde gesagt, fragt die Tastatur ab. Diese Formulierung war jedoch nicht ganz richtig: KM READ KEY hat genaugenommen mit der Abfrage der Tastatur gar nichts zu tun. Diese wird von einer, für den Benutzer "transparenten", Routine im Zuge der Interruptbehandlung erledigt. Diese "Update Key State Map" genannte Routine fragt die Tastaturmatrix mit Hilfe einer Routine im Machine Pack ab und bereitet die gewonnenen Rohdaten auf. Für jede neu gedrückte Taste wird ein Eintrag in einem Ringbuffer gemacht. Der Ringbuffer ist ein Zwischenspeicher, der eine FIFO-Struktur darstellt. First In-First Out, heißt, daß der Eintrag, der als erstes in den Ringbuffer geschrieben wurde auch als erstes aus ihm ausgelesen wird. Anders dagegen beim Stack, der klassischen LIFO-Struktur (Last In-First Out), aus dem die Einträge in genau der umgekehrten Reihenfolge gelesen werden, in der sie in den Stack geschrieben wurden. Ein solcher Eintrag in den Ringbuffer wurde von uns als die Tastenkoordinaten der entsprechneden Taste bezeichnet: Er enthält sowohl die Nummer der Tastaturzeile in einem Byte, als auch das der Taste entsprechende Bit, isoliert im zweiten Byte. Der Ringbuffer kann 20 Einträge aufnehmen, d.h. es können bis zu 20 Tasten "vorgetippt" werden, ohne das die Tastatur abgefragt werde mußte. Da ein Eintrag jeweils zwei Byte lang ist, folgt daraus, daß der Ringbuffer eine Länge von insgesamt

40 Byte hat - zuzüglich der diversen Zeiger (Schreib- und Lesezeiger) und Verwaltungsbytes.

Wie bereits oben angedeutet, läuft dieser Prozeß, gesteuert durch den Interrupt, asynchron zum Ablauf der Programme etwa mit einer Frequenz von 50 Hz, also fünfzigmal in der Sekunde ab. KM READ KEY holt sich, wenn es aufgerufen wird, den ersten, d.h. auch den ältesten Eintrag aus dem Ringbuffer und berechnet aus diesen Tastenkoordinaten die Nummer der Taste. Dann wird aus einer Tabelle der Code der entsprechenden Taste gelesen. Dabei ist zu beachten, daß der ASCII-Code einer Taste nicht konstant ist. Einer Taste sind drei verschiedene Codes zugeordnet, je nachdem ob ConTRoL, SHIFT oder keine der beiden gedrückt ist. Dementsprechend gibt es auch drei verschiedene Tabellen: die Key Translation Tabelle für den "Normalfall", die Key Ctrl Tabelle für den CTRL-Code der Taste und die Key Shift Tabelle für den SHIFT-Code der Taste. Die Umwandlung der Tastennummer in den zugeordneten ASCII-Code geschieht dann durch die drei Routinen KM GET TRANSLATE, KM GET SHIFT und KM GET CTRL. Welche dieser Routinen von KM READ KEY zur Ermittlung des Tastencodes herangezogen wird, wird anhand der SHIFT- und CTRL-Flags entschieden, die immer zusammen mit den Tastenkoordinaten im Ringbuffer abgelegt werden.

3.7.3 Die Behandlung eines Breaks

Neben den auf diese Weise entstandenen Tastencodes gibt es einen Code mit einer besonderen Bedeutung: den Code $EF bzw. 239. Dieses Zeichen haben wir das BRK-Zeichen getauft, weil es in engem Zusammenhang mit der Sonderstellung der ESC-Taste und der Break-Bearbeitung im Keyboard Manager steht. In der Routine Update Key State Map, auf der untersten Ebene während eines Interrupts also, wird gesondert geprüft, ob außer TAB noch eine Taste in der achten Zeile (in der auch ESC liegt) gedrückt ist. Ist dies der Fall, so wird nach beendeter Abfrage der Tastatur und nachdem alle Tasten in den Ringbuffer eingetragen worden sind, in eine Routine namens KM TEST BREAK gesprungen. Diese prüft, ob CTRL-SHIFT-ESC gedrückt und einen Reset auslöst ist. Wenn ansonsten ESC gedrückt ist, geht KM TEST BREAK in eine Routine mit dem Namen KM BREAK EVENT, die den Break Event aktiviert, falls er erlaubt ist. Als Zeichen für die Aktivierung des Break Events schreibt KM BREAK EVENT ein $EF in den Ringbuffer. Dieses $EF, das ja eigentlich nicht der Tastencode sondern nur ein Teil der Tastenkoordinaten darstellt, wird von KM READ KEY als Sonderfall erkannt und als Tastencode übernommen. Das BRK-Zeichen ist also nicht mit dem ESC-Zeichen gleichzusetzen, dessen Code entgegen dem ASCII-Standard ja $FC ist. Es zeigt die Aktivierung des Events an, während das ESC-Zeichen völlig unabhängig davon

ausgegeben wird. Auch kann das ESC-Zeichen jeder beliebigen Taste zugeordnet werden, wohingegen das BRK an die ESC-Taste gebunden ist.

Der Break Event selber wird vom Keyboard Manager verwaltet. Der User bzw. das Basic, das wir hier als "User" des Operating Systems verstehen, muß der Routine KM ARM BREAK lediglich die Adresse der Routine, die er zur Behandlung eines Breaks vorsieht, und die entsprechende ROM-Konfiguration übergeben. KM ARM BREAK baut dann den synchronen Event Block auf und "erlaubt" den Event. Diese Routine setzt also das Flag, das der Routine KM BREAK EVENT angibt, ob es den Event Block überhaupt einhängen darf oder nicht - z.B. wenn der Event Block undefiniert ist oder der User den Break Event nicht eingeschaltet haben möchte. Die Routine KM DISARM BREAK schaltet den Break Event wieder aus, indem sie das Flag wieder zurücksetzt und ihn für den Fall aushängt, daß der Event gerade eingehängt ist.

3.7.4 Einflußmöglichkeiten des Benutzers

Der Benutzer hat viele Möglichkeiten, auf die Abläufe im Keyboard Manager steuernd Einfluß zu nehmen. Lobenswert hinsichtlich des Schneider-Basics ist, daß nahezu alle Features des Tastatur-Verwaltungsprogrammes auf seiner Ebene "herausgeführt" sind. Somit können sie vom Basic-Programmierer, und nicht nur von Maschinensprache aus, benutzt werden. Dazu gehören

1. die Definition von Expansion Strings, die mit der Routine KM SET EXPAND geschieht,

2. das Setzen der Verzögerungen für die Tastenwiederholung, sowohl der 1. Verzögerung als auch der Repeat-Verzögerung, mit der Routine KM SET DELAY,

3. die Möglichkeit, über das Ein- bzw. Ausschalten des Repeats für jede einzelne Taste bestimmen zu können. Hierfür ist die Routine KM SET REPEAT vorgesehen.

4. Der Benutzer kann ferner den Tasten der Tastatur völlig wahlfrei ASCII-Codes zuordnen, so daß die Bedeutung der einzelnen Tasten und deren Festlegung dem Benutzer überlassen bleibt. Der Keyboard Manager verfügt zu diesem Zweck über die Routinen KM SET TRANSLATE, KM SET SHIFT und KM SET CTRL.

5. Im CPC 664 und im CPC 6128 gibt es zusätzlich noch die Möglichkeit, die Caps Lock und Shift Lock Flags per Programm zu setzen.

Die entsprechende Routine im Keyboard Manager trägt den Namen KM SET LOCKS.

6. Auch nur im CPC 664/6128 gibt es die Routine KM FLUSH. Sie dient dazu, alle Tastenkoordinaten, die sich zu der Zeit im Ringbuffer befinden, zu löschen, d.h. alle gedrückten Tasten zu löschen.

3.7.5 Ausgaberoutinen des Keyboard Managers

Unter "Ausgaberoutinen" sind in diesem Zusammenhang Unterprogramme zu verstehen, die einem übergeordneten, aufrufenden Programm Informationen übergeben. Sie dienen vorzugsweise also nicht zum Setzen bestimmter Informationen, wie die Routinen unter 3.7.4. Die wichtigsten haben wir bereits kennengelernt: KM READ CHAR zum Holen eines allgemeinen Zeichens, KM WAIT CHAR zum Warten auf ein solches Zeichen. Ferner hatten wir schon KM READ KEY besprochen, die eine Taste von der Tastatur holt und entsprechend auch KM WAIT KEY, die auf eine solche Taste wartet. Daneben kann der Benutzer jedoch noch andere Informationen aus dem Keyboard Manager beziehen:

1. **KM GET STATE** holt den augenblicklichen Status von Shift Lock und Caps Lock.

2. **KM GET JOYSTICK** holt die Stellung der beiden Joysticks.

3. **KM GET DELAY** holt die Verzögerungs-Werte, die momentan gesetzt sind.

4. **KM GET REPEAT** gibt an, ob für eine Taste die Tastenwiederholung eingeschaltet ist oder nicht.

5. **KM TEST KEY** gibt für eine Taste an, ob sie gedrückt ist, und gibt auch die Ctrl/Shift-Flags aus.

3.7.6 Das Initialisieren des Keyboard Managers

Um den Keyboard Manger vollständig in den Ausgangszustand zurückzusetzen, muß man die Routine KM INITIALIZE anspringen. Sie setzt einzelne Parameter zurück, setzt die ASCII-Tabellen auf deren Ausgangswerte zurück, initialisiert die Tabellen mit den Zeilenrückmeldungen, löscht den Ringbuffer und den Put Back Buffer und setzt den Expansion String Buffer auf seine Default Strings.

Wenn der User jedoch nicht alles neu initialisieren möchte, dann kann er z.B. auch KM RESET anspringen, die lediglich den Ringbuffer und den Put Back Buffer löscht, den Expansion Buffer zurücksetzt und den Break Event ausschaltet. Dies bietet sich an, wenn man sich eigene Tastenbelegungen definiert hat, die man nicht vernichten möchte, den Keyboard Manager aber dennoch in einen möglichst definierten Ausgangszustand bringen will. Hat man jedoch auch Expansion Strings definiert, die man erhalten möchte, und sei es nur zum Löschen aller sich momentan im Ringbuffer befindliche Zeichen, so muß man die Routine "Ringbuffer initialisieren" benutzen. Dabei sollte man jedoch darauf achten, gleichzeitig auch den Break Event zu "disarmen". Es ist nämlich zu bedenken, daß sich im Ringbuffer eventuell ein BRK-Zeichen befindet, das ja als Kennzeichen für einen eingehängten Break Event dient. Wird es gelöscht, so hat das User-Programm keinen Hinweis mehr darauf, daß ein Break Event in der Pending Queue auf seine Ausführung wartet. Der alte Event verhindert dann auch das Einhängen neuer Events.

3.8 Der SOUND MANAGER (SOUND)

3.8.1 Allgemeines

Der Sound Manager ist der Teil des CPC-Betriebssystems, der die Verwaltung der Sound-Ausgabe auf dem Schneider-Computer übernimmt. Er ist ein sehr nützliches Stück Software mit erstaunlichen Features. So verwaltet er z.B. alle drei von der Hardware unterstützten Kanäle, jeweils mit eigenen, frei programmierbaren Hüllkurven - und zwar Hüllkurven sowohl für die Tonhöhe wie auch für die Lautstärke. Weiterhin richtet er für jeden Kanal eine Warteschlange ein, so daß mehrere Töne hintereinander ohne Wartezeiten an einen Kanal übergeben werden können, wodurch ein relativ verzögerungsfreier Betrieb ermöglicht wird.

Man kann den Sound Manager als ein Programm betrachten, das dem klangerzeugenden Baustein AY 3-8912 sozusagen "aufgesetzt" wurde. Als ein solches leistet er einiges: Er unterstützt sämtliche Eigenschaften des Sound Chips und trägt dazu bei, dessen Unzulänglichkeiten auszubügeln, wie z.B. die Beschränkungen hinsichtlich der Hüllkurven.

3.8.2 Die Verwaltung der Kanäle

Wie bereits oben gesagt, verwaltet der Sound Manager alle drei Kanäle völlig autonom und voneinander unabhängig. Damit er zu jeder Zeit über den Zustand eines bestimmten Kanals informiert ist, sind sämtliche Daten, die diesen Zustand beschreiben, für jeden Kanal in einem sogenannten Parameter-Block zusammengefaßt. Diese Parameter sind in zwei Hauptteile gegliedert: 1. in den aktuellen Status der gerade laufenden Tones und 2. in die restliche Warteschlange, in der maximal vier weitere Töne pro Kanal abgelegt werden können. Diese vier Plätze in den Params sind in Form eines Ringbuffers organisiert. Im folgenden wollen wir uns den Aufbau eines solchen Parameter-Blockes näher ansehen:

Byte	Funktion
00	Kanalnummer, 0-2
01	Kanalmaske, b0, b1 oder b2 gesetzt
02	Rauschmaske, b3, b4 oder b5 gesetzt
03	Kanalstatus: b4=1: Kanal aktiv b3=1: Warteschlange im Haltezustand b2-b0: Rendezvous-Status

04	ENT-Flag: b7=1: ENT-Folge schwebend b0=1: ENT-Folge muß bedient werden
05	laufende Pausenzeit für ENT
06	laufende Pausenzeit für ENV
07	ENV-Flag b7=1: ENV-Folge schwebend b0=1: ENV-Folge muß bedient werden
08/09	Tonlänge (noch verbleibend)
0A/0B	Anfangsadresse der laufenden ENV-Folge
0C	Anzahl der ENV-Gruppen
0D/0E	Adresse der laufenden ENV-Gruppe
0F	laufende Lautstärke
10	laufende Schrittzähler für ENV
11/12	Anfangsadresse der laufenden ENT-Folge
13	Anzahl der ENT-Gruppen
14/15	Adresse der laufenden ENT-Gruppe
16/17	laufenden laufenden Periodendauer
18	laufende Schrittzähler für ENT
19	Nummer des nächsten Eintrags in Warteschlange
1A	Anzahl der Blöcke in der Warteschlange
1B	Nummer des nächsten freien Blockes
1C	Anzahl der freien Datenblöcke
1D/1E	Adresse des synchronen Events
1F-26	Datenblock #1
27-2E	Datenblock #2
2F-36	Datenblock #3
37-3E	Datenblock #4

Wie man sieht, ist ein solcher Parameterblock genau 63 Byte lang. Im CPC existieren drei solche Blöcke, die im Speicher exakt hintereinander liegen. Neben der oben gemachten Unterteilung in den aktuellen Zustand (00-1F) und die verbleibende Warteschlange (20-3F) läßt sich der erste Teil des Blockes noch feiner gliedern:

1. **Die drei Kanalkennungen (00-02):** Neben der Nummer des Kanals ist die Kanalmaske, in der bei jedem Kanal das entsprechende Bit gesetzt ist, u.a. dazu da, bei bestimmten Routinen als Vergleichsbyte für die übergebenen Kanalbits zu dienen. Die Kanalmaske wird auch zum Aktivieren des Kanals im Kontrollregister des PSG verwendet. Die Rauschmaske dient zur Aktivierung des Rauschens für den entsprechenden Kanal im PSG.

2. **Die allgemeine Verwaltung (03-09):** Sie dienen dazu, festzustellen, ob der Kanal vom Sound Event bedient werden muß, d.h. ob eine ENV- oder ENT-Folge fortgesetzt werden muß, bzw. ob der Ton noch läuft oder ob der Kanal mit anderen Kanälen synchronisiert ist ("Rendezvous").

3. **Die ENV-Verwaltung (0A-10):** In diesem Bereich liegt alles, was zur Bearbeitung der Lautstärke-Hüllkurve bzw. zur allgemeinen Verwaltung der Lautstärke nötig ist.

4. **Die ENT-Verwaltung (11-18):** Dieser Bereich enthält sämtliche Parameter zur Bearbeitung der Ton-Hüllkurven.

5. **Verwaltung der Warteschlange (19-1E):** Hier liegen die Daten, die zur Koordination der Blöcke in den vier Plätzen der Warteschlange benötigt werden.

Im folgenden werden wir zunächst auf die Bearbeitung einer ENV/ENT-Folge eingehen, dann die Abarbeitung eines ganzen Tons betrachten und uns anschauen, wie eine Warteschlange behandelt wird.

3.8.3 Die Abarbeitung einer ENV/ENT-Folge

Auf die Bedeutung der ENV- und ENT-Hüllkurven, sowie auf bestimmte Fachbegriffe, wie z.B. "Hüllkurve", können wir im Rahmen dieses Buches nicht näher eingehen. Im Handbuch zum CPC existiert dazu jedoch ein recht gutes Kapitel, das wir an dieser Stelle voraussetzen müssen.

3.8.3.1 Das Format einer ENT-Folge

Wie Sie sicher wissen, kann man von Basic aus 15 verschiedene ENT-Folgen definieren. Man übergibt dem entsprechenden Befehl (ENT) die Nummer der Folge, die man definieren möchte und maximal fünf Parametergruppen mit je zwei oder drei Parametern. Soll die Folge nach Beendigung wiederholt werden übergibt man, zusammen mit einem Flag, eine negative Nummer.

Unterhalb der Basic-Ebene, d.h. konkret bei der Übergabe der dem Basic übergebenen Parameter zum Betriebssystem, ist der Aufbau einer ENT-Folge nur sehr leicht verändert. Die Parameter sind jetzt in einen Block von genau 16 Byte zusamengefaßt: Das erste Byte ist eine Art "Kopfbyte" für die Folge, es enthält die Anzahl der Parametergruppen und das Wiederholungsflag. Die anderen 15 Byte stellen die fünf Parametergruppen dar, wobei für jede Gruppe genau drei Byte benötigt werden. Der Aufbau einer ENT-Folge sieht im Überblick also so aus:

Byte	Funktion
0	b7=1: ENT-Folge wird wiederholt b6-b0: Anzahl der Parametergruppen
3n+1	Schrittanzahl < $F0
3n+2	Schrittweite (2er Komplement)
3n+3	Pausenzeit

beziehungsweise:

Byte	Funktion
3n+1	b7-b4=$F als Kennzeichen b3-b0: Tonperiode, oberstes Nibble
3n+2	Tonperiode, Lo-Byte
3n+3	Pausenzeit

In der Tabelle oben ist n die Nummer der Parametergruppe (0..4). Wie zu sehen ist, gibt es zwei verschiedene Arten von Gruppen. In der ersten Form wird dem Sound Manager die Veränderung der Tonperiode übergeben, also eine Art "relativer" Tonperiode. In der zweiten Form wird die Tonperiode - so, wie sie in den PSG geschrieben wird - direkt übergeben. Beide Formen sind natürlich innerhalb einer Folge beliebig mischbar.

3.8.3.2 Das Format einer ENV-Folge

Eine ENV-Folge sieht in ihrem Aufbau einer ENT-Folge zunächst einmal sehr ähnlich: auch sie verfügt über ein Kopfbyte und auch in der ENV-Folge gibt es fünf Parametergruppen mit jeweils drei Byte. Dies ist eigentlich auch kein Wunder, da die Programmierung von Basic aus sehr ähnlich aussieht. Betrachten wir nun das Format einer solchen ENV-Folge:

Byte	Funktion
0	Anzahl der Parametergruppen
3n+1	b7=0 als Flag für erste Form b6-b0: Schrittzahl < $80
3n+2	Schrittweite (2er Komplement)
3n+3	Pausenzeit

beziehungsweise:

Byte	Funktion
3n+1	b7=1 als Flag für zweite Form b3-b0: Registerwert für PSG-Hüllkurve
3n+2	Hüllkurvenperiode, high
3n+3	Hüllkurvenperiode, low

Auch in dieser Tabelle meint n die Nummer der entsprechenden Gruppe (0..4). Wie auch schon für ENT-Gruppen, so gibt es für ENV-Gruppen

ebenfalls zwei Formen: die Angabe einer eigenen Veränderung der Lautstärke durch eine selbstdefinierte Veränderung der Hüllkurve (erste Form) bzw. die Spezifizierung einer PSG-Hüllkurve (siehe Kapitel 1.5), zusammen mit der entsprechenden Periodendauer für deren Veränderung.

3.8.3.3 Das Definieren einer Hüllkurve

Trotz der Unterschiede im Aufbau der ENV- und der ENT-Hüllkurven ist das Definieren von solchen Folgen im wesentlichen identisch. Dies erkennt man schon auf Basic-Ebene: die Syntax für die Definition ist bei beiden dafür vorgesehenen Befehlen (ENV und ENT) fast genau gleich. Auf Maschinenebene ist diese Ähnlichkeit noch extremer: Die Folgen werden nunmehr nur noch als 16-Byte-Blöcke behandelt; der Inhalt dieser Blöcke ist für die Definition ohne jede Bedeutung.

Die Definition selber geschieht über zwei Routinen des Sound Managers: SOUND AMPL ENVELOPE (für ENV-Folgen) und SOUND TONE ENVELOPE (für ENT-Folgen). Das benutzende Programm übergibt diesen Routinen die Nummer der Hüllkurve, die definiert werden soll, und die Adresse des 16-Byte-Blocks, der die Definition darstellt. Die Routinen kopieren dann den Block in eine Tabelle an die Stelle, die der Nummer entspricht. Es gibt jeweils eine Tabelle für ENV- und eine für ENT-Folgen und die Nummer muß im Bereich von 1 bis inklusive 15 liegen. Von nun an kann nur über die Nummer auf eine bestimmte Hülkurve zugegriffen werden.

3.8.3.4 Die Abarbeitung der Folgen

Auch bei der Bearbeitung der Folgen lassen sich Ähnlichkeiten feststellen. Wegen der vollständig unterschiedlichen Semantik der beiden Hüllkurven, und der damit verbundenen unterschiedlichen Behandlung, läuft sie natürlich auf völlig verschiedenen Wegen ab. Die Routinen, die die Folgen behandeln, sind also vollständig unterschiedlich.

Wie so vieles im CPC, so läuft auch die Verwaltung des Sounds unter Interruptsteuerung. Dies geschieht derart, daß bei jedem dritten Interrupt (300 Hz / 3 = 100 Hz, d.h. alle 0,01 s) die Routine "Scan Sound Queues" nachschaut, ob in den drei Kanälen etwas getan werden muß und dann die entsprechende Behandlung einleitet. Scan Sound Queues schaut nach, indem es den Pausenzähler der laufenden Gruppe erniedrigt und, wenn er null ist, ein Flag testet. Dieses Flag gibt an, ob die entsprechende Folge in dem Kanal überhaupt aktiviert ist. Wenn sie aktiviert war, so setzt die Routine ein Flag, das angibt, daß eine der Folgen behandelt werden muß. Diesen Prozeß macht Scan Sound Queues mit jedem Kanal durch - pro Kanal je-

weils mit der ENV- und der ENT-Folge. Das Ergebnis dieses Durchsuchens ist schließlich ein Flag, das anzeigt, ob irgendeine Folge behandelt werden muß. Ist dies der Fall, so wird der asynchrone Sound Event eingehängt, der dann die Bearbeitung der Kanäle übernimmt.

Der Sound Event sucht in allen Kanal-Blöcken nach ENV/ENT-Folgen, die bedient werden müssen. Die Bearbeitung einer solchen Folge sieht so aus, daß der Schrittzähler erniedrigt, der Offset zum jeweiligen Parameter addiert (bei ENV ist es die Lautstärke, bei ENT die Periodendauer) und schließlich die Pausenzeit neu gesetzt wird. Bei Gruppen, die die zweite Form einer ENV- oder ENT-Folge besitzen, wird der entsprechende Wert des Parameters natürlich anders ermittelt. Wenn der Schrittzähler Null war, d.h. wenn die laufende Gruppe zu Ende ist, so wird die nächste Gruppe geholt und gesetzt.

3.8.4 Die Abarbeitung der einzelnen Töne

Der Prozeß, der der Bearbeitung der Gruppen innerhalb eines Tons übergeordnet ist, ist die Bearbeitung der verschiedenen Töne innerhalb der Warteschlange eines Kanals (der Kürze halber auch "Queue" genannt). Wir hatten schon erwähnt, daß die Warteschlange neben dem gerade aktiven Ton auch noch vier weitere Töne aufnehmen kann und daß sie als Ringbuffer realisiert ist. Im folgenden werden wir uns diese Struktur ein wenig genauer betrachten.

3.8.4.1 Der Aufbau eines Datenblockes

Im Abschnitt über den Aufbau eines Kanal-Blockes hatten wir bereits die Bereiche, in denen die Daten für die jeweils wartenden Töne abgelegt sind, als die Datenblöcke 1 bis 4 gekennzeichnet, ihre innere Struktur jedoch nicht weiter aufgeführt. Dies wollen wir an dieser Stelle nun tun:

Byte	Funktion
0	Datenstatus b7=1: Flush, Kanal-Params löschen b3=1: Hold, Warteschlange in Haltezustand b2-b0: Rendezvous-Status des Tons
1	b7-b4: Nummer der ENV-Folge b3-b0: Nummer der ENT-Folge
2/3	Tonperiode, Startwert
4	Rauschperiode
5	Lautstärke, Startwert
6/7	Tondauer

Es ist durchaus kein Wunder, wenn Ihnen diese Parameter bekannt vorkommen: sie entsprechen ziemlich genau denen, die dem Befehl SOUND in Basic übergeben werden. Diesem Befehl muß natürlich noch angegeben werden, an welchen Kanal der Ton geschickt werden soll. Die Funktion der einzelnen Bytes dürfte weitgehend klar sein. Ist dies nicht der Fall, so schlagen Sie bitte im entsprechenden Kapitel Ihres Basic-Handbuches nach, wo der Basic-Befehl SOUND sowie die zu übergebenden Parameter genau beschrieben sind.

3.8.4.2 Die Übergabe eines Tons an den Sound Manager

Die Übergabe eines Tons geschieht mit der Routine SOUND QUEUE. Diese Routine bekommt einen neun Byte langen Datenblock übergeben, der annähernd die Form hat, die auch ein Datenblock in den Kanal-Params hat. In diesen Daten ist z.B. auch der Kanal enthalten, auf den der Ton ausgegeben werden soll. Ist in diesem Kanal noch ein Platz in dessen Warteschlange frei, so werden die Daten noch ein wenig komprimiert und dann in den nächsten freien Datenblock geschrieben. Hier wird auch die Struktur des Ringbuffers sichtbar (siehe Abschnitt 2.2.2 über FIFOs): die Töne werden hinten an die Schlange angehängt und vorne wieder herausgenommen.

3.8.4.3 Die Bearbeitung eines Tons

Wenn der Sound Event auf das Ende eines Tons stößt, so schaut er nach, ob in der Warteschlange noch Töne auf ihre Ausgabe warten. Ist dies der Fall, so wird der Ton "geholt", d.h. die Daten im Datenblock werden in den Hauptblock geschrieben. Die Nummern der ENV- und der ENT-Folgen werden jedoch nicht in den Hauptblock gebracht. Aus diesen Nummern werden vielmehr die Adressen der Folgen berechnet, mit denen dann in der Verwaltung des Hauptblockes gearbeitet wird. Es existiert für jede Folge jeweils ein Zeiger auf den Anfang der Folge und ein Zeiger in die momentan laufende Folgen-Gruppe hinein. Diese Art, die Daten aufzugliedern ermöglicht eine schnellere Auswertung des Hauptblockes und somit eine schnellere Bearbeitung des Sound Events.

3.8.5 Der Begriff "Aktivität"

Im ROM Listing taucht beim Sound Manager häufiger der Begriff "Aktivität" auf, weshalb wir ihm hier einmal kurz erläutern sollten.

Die Aktivitäten stehen in engem Zusammenhang mit der Bearbeitung eines Tons. Sie sind in einem Byte abgelegt, von dem jeweils nur die unteren drei Bits benötigt werden. Jedes Bit stellt dabei den Zustand des

entsprechenden Kanals dar, gibt also an, ob der Kanal an- oder ausgeschaltet ist. Eine Eins markiert den Kanal als "aktiv", eine Null bedeutet, daß der Kanal ausgeschaltet ist.

Im CPC gibt es zwei Aktivitätsbytes, die von besonderer Bedeutung sind: eines haben wir "laufende Aktivitäten", das andere "alte Aktivitäten" genannt. Die laufenden Aktivitäten geben an, welche Kanäle im Moment gerade aktiviert sind. Die alten Aktivitäten dienen zum "Einfrieren" und Wiederaufnehmen der Tonausgabe mit SOUND HOLD und SOUND RELEASE.

3.8.6 Events im Sound Manager

Wie einige andere Packs auch, verfügt der Sound Manager über eine Event-Bearbeitung. Der Stellenwert, den Events im Sound Manager einnehmen ist recht beachtlich: Die zentrale Routine des Sound Managers ist immerhin der "Sound Event". Wir wollen uns in diesem Abschnitt jedoch nicht mit diesem Event beschäftigen; vielmehr interessieren uns die synchronen Events, die den einzelnen Kanälen zugeordnet sind, und die Sie von Basic aus über ON SQ ansprechen können.

Wie aus dem Basic bekannt, wird der mit ON SQ GOSUB angegebene Programmteil immer dann ausgeführt, wenn in der Queue des entsprechenden Kanals ein Platz frei wird. Das Basic verwaltet das derart, daß es die Adresse des Events an die Routine Sound Arm Event übergibt. Jeder Kanal verfügt über einen eigenen, natürlich synchronen Event Block, obschon die Event-Routine dieselbe ist. Der Sound Arm Event prüft, ob noch Platz in der Warteschlange des Kanals ist. Ist dies der Fall, so wird der Event gleich von dieser Routine eingehängt, andernfalls in einem dafür vorgesehenen Platz im Parameter-Block des Kanals abgelegt. Die Routine, die den nächsten Eintrag in der Warteschlange auswertet, hängt dann den Event, wenn vorhanden, auch gleich ein.

3.8.7 Die Routinen des Sound Managers

Im vorangegangenen Text haben wir bereits einige der Routinen des Sound Managers namentlich erwähnt. An dieser Stelle nun möchten wir einen kurzen Überblick über die wichtigsten Routinen und ihre Funktionen geben.

SOUND RESET: Er setzt alle mit der Sound-Ausgabe verbundenen Systeme zurück, d.h. schaltet den Sound im PSG aus, initialisiert die Parameter-Blöcke und den übrigen Arbeitsspeicher des Sound Managers.

SOUND HOLD: Diese Routine setzt die laufenden Aktivitäten als alte und löscht die laufenden, d.h. friert die Tonausgabe ein.

SOUND CONTINUE: Kehrt die Funktion von Sound Hold um, indem alle alten Aktivitäten als laufende gesetzt und die entsprechenden Kanäle aktiviert werden.

Sound Event: Dies ist die Schlüsselroutine im Sound Manager. Wenn "Scan Sound Queues" festgestellt hat, daß in einem der Kanäle eine Folge bearbeitet werden muß, so hängt sie diesen Event in die Asynchronous Pending Queue ein. Diese Routine steuert dann die Bearbeitung der ENV/ENT-Folgen im CPC.

Scan Sound Queues schaut nach, ob eine der Folgen in irgendeinem der z.Z. aktiven Kanäle eine Bearbeitung erfordert und hängt den Sound Event ein. Sie wird durch den Interrupt mit einer Frequenz von 300 Hz aufgerufen, geht aber nur jedes dritte Mal (100 Hz) an die Bearbeitung der Folgen.

SOUND QUEUE: Sie ist für den Benutzer die wohl bedeutendste Routine. Mit ihr kann man dem Sound Manager Töne übergeben und sie zur Ausgabe an einen bestimmten Kanal schicken. Sie baut dann dort - wenn vom Platz her möglich - einen Eintrag in der Warteschlange des angegebenen Kanals auf, in den sie die übergebenen Parameter schreibt.

SOUND RELEASE: Diese Routine ermöglicht es dem Benutzer, den Haltezustand einer Queue in den angegebenen Kanälen auszuschalten und die Bearbeitung des Kanals erneut zu aktivieren. Eine Warteschlange wird durch entsprechende Übergabeparameter bei Sound Queue in den Haltezustand versetzt. Dieses Einfrieren einer Queue ist nicht zu verwechseln mit dem Einfrieren der Sound Ausgabe durch Sound Hold. Aus eigener Erfahrung weist der Autor darauf hin, daß man sich diesen Unterschied noch einmal genau klar machen sollte.

SOUND CHECK: Diese Routine gibt den Status des Kanals und die Anzahl der freien Plätze in der Warteschlange des angegebenen Kanals zurück. Auch wird der dem Kanal zugeordnete synchrone Event gelöscht.

SOUND ARM EVENT legt im Parameter-Block des angegebenen Kanals die übergebene Adresse des entsprechenden Kanal-Events ab, der immer dann eingehängt wird, wenn sich ein freier Platz in der Warteschlange des Kanals ergibt. Sound Arm Event prüft vorher ab, ob bereits ein Platz in der Queue frei ist und hängt den Event dann gleich ein.

SOUND AMPL ENVELOPE: Mit dieser Routine ist es dem Benutzer möglich, eine ENV-Hüllkurve unter einer Nummer von 1 bis 15 zu definieren,

die dann unter dieser Nummer als Parameter bei der Übergabe von Tönen mittels Sound Queue mitübergeben werden kann.

SOUND TONE ENVELOPE: Diese Routine ist das Gegenstück zu Sound Ampl Envelope für die Definition von ENT-Hüllkurven.

SOUND A ADDRESS holt die Tabellenadresse der durch die Nummer angegebenen ENV-Hüllkurve.

SOUND T ADDRESS holt die Tabellenadresse der durch die Nummer angegebenen ENT-Hüllkurve.

3.9 Der CASSETTE MANAGER (CAS)

3.9.1 Aufgaben des Packs

Dieses Pack verwaltet maximal ein Ein- und ein Ausgabefile gleichzeitig. Die zur Fileverwaltung benötigten Routinen werden direkt vom Basic aufgerufen. Falls eine Diskettenstation an den CPC angeschlossen ist, werden die Aufrufe des Basics entsprechend umgeleitet. Die Bedeutungen der Routinen bleiben sonst die gleichen.

Mit der Routine CAS INITIALIZE werden Ein- und Ausgabe abgebrochen sowie die Baudrate und das Meldungs-Flag werden initialisiert. Das Meldungs-Flag gibt an, ob während des Ladens und Speicherns von Programmen entsprechende Meldungen ausgegeben werden sollen. Es kann mit der Routine CAS NOISY gesetzt werden. Basic setzt dieses Flag je nachdem, ob ein Ausrufezeichen den Filenamen anführt oder nicht.

3.9.2 Aufbau eines Files

Jedes File, das auf Kassette gespeichert wird, egal ob Programm- oder Daten-File, wird in Blöcke von 2 KByte Größe aufgeteilt. Jeder dieser Blöcke besteht wiederum aus einem 64 Byte langen Header, der Informationen über den Block enthält, und den 2 KByte eigentlichen Daten. Ein Header eines Blocks ist wie folgt aufgebaut:

Byte(s)		Bedeutung
hex	dez	
00-0F	00-15	Filename, ggf. aufgefüllt mit Null-Bytes
10	16	Nummer des Blocks
11	17	Zeichen für letzten Block im File, sonst = 0
12	18	Filetyp \$00 = Basic-Programm \$01 = geschütztes Basic-Programm \$02 = Maschinenprogramm \$16 = ASCII-Datei
13-14	19-20	Länge des Blocks
15-16	21-22	Ladeadresse des Blocks
17	23	Zeichen für ersten Block im File, sonst = 0
18-19	24-25	Länge des gesamten Files
1A-1B	26-27	Aufrufadresse des Maschinenprogramms
1C-3F	28-63	unbenutzt

3.9.3 Die Verwaltung von Files

Mit der Routine CAS OUT OPEN kann ein Ausgabefile eröffnet werden. Der Routine muß neben den Parametern des Filenamens auch die Adresse eines 2 KByte großen Bufferbereichs zur Zwischenspeicherung eines Blocks übergeben werden. CAS OUT OPEN überträgt den Filenamen in einen Buffer für den Header und setzt den Filetyp auf $16 (ASCII-Datei). Der Filetyp kann gegebenenfalls später noch geändert werden. Auf Kassette schreibt diese Routine nichts. Es wird jedoch ein Bereich für den File-Status und die Bufferzeiger initialisiert:

Byte	Bedeutung		
0	File-Status		
	464	664/6128	
	0	0	= nicht geöffnet
	1	1	= gerade eröffnet
	2	5	= zeichenweise Datei (ASCII-Datei)
	3	2	= Direkt-Datei (Programm)
	4	3	= Abbruch durch ESC
	5	4	= CAS CATALOG aktiv
1/2	Adresse des Buffers		
3/4	laufender Bufferzeiger		
ab 5	Buffer für Blockheader		

CAS OUT OPEN setzt den File-Status auf "1". Wenn der Status vorher nicht "0" war, das File also schon eröffnet wurde, gibt CAS OUT OPEN einen Fehler (CY=0) zurück.

Die Routine CAS OUT CHAR legt bei ihrem ersten Aufruf den File-Status und in der Regel auch den (voreingestellten) Filetyp auf eine zeichenweise (ASCII-)Datei fest. Es muß nun Zeichen für Zeichen an diese Routine übergeben werden. CAS OUT CHAR speichert die übergebenen Zeichen im Ausgabebuffer zwischen und schreibt dann einen ganzen 2-KByte-Block auf einmal auf Band. CAS OUT CLOSE schließt das File ordnungsgemäß. Der aktuelle Bufferinhalt wird auf Band geschrieben, auch wenn der Buffer nicht ganz voll sein sollte. CAS OUT ABANDON dagegen bricht die Ausgabeoperation sofort ab.

Mit der Routine CAS OUT DIRECT läßt sich, z.B. für Programme, ein Speicherbereich direkt auf Kassette schreiben. Der Bereich wird allerdings trotzdem in 2-KByte-Blöcke aufgeteilt. Durch CAS OUT CLOSE wird der letzte Block auf Band geschrieben und das File geschlossen. Die Routinen CAS OUT CHAR und CAS OUT DIRECT können nicht miteinander kombiniert eingesetzt werden - andernfalls wird ein Statusfehler ausgegeben.

Ebenso wie für die Ausgabe gibt es für die Eingabe von Kassette die Routinen CAS IN OPEN, CAS IN CHAR, CAS IN DIRECT, CAS IN CLOSE und CAS IN ABANDON. Ein Unterschied zur Ausgabe besteht darin, daß CAS IN OPEN schon einen Block von Kassette liest und nach einem Block sucht, dessen Filename mit dem des gesuchten Headers übereinstimmt. Zu diesem Zweck existieren übrigens zwei Eingabe-Header-Buffer: einer für den gesuchten und einer für den gelesenen Header.

Es ist ohne weiteres möglich, ein Programm, das mit CAS OUT DIRECT quasi in einem Stück auf Band geschrieben wurde, zeichenweise mit CAS IN CHAR wieder einzulesen. Die Verwendung von CAS IN CHAR bzw. CAS OUT CHAR bedeutet also nicht automatisch, daß eine ASCII-Datei bearbeitet wird. Was jedoch, wie auch bei CAS OUT CHAR und CAS OUT DIRECT, nicht funktioniert, ist das Einlesen des restlichen Files mit CAS IN DIRECT, nachdem ein Teil des Files bereits mit CAS IN CHAR eingelesen wurde - oder umgekehrt.

Die Routine CAS RETURN setzt Bufferzeiger und -länge auf das zuletzt eingelesene Zeichen, das dadurch noch einmal eingelesen wird. Dies kann nützlich sein, wenn man bestimmte Sonderfälle abfangen will. So muß z.B. bei einem Linefeed, das auf ein Carriage Return folgt, das nächste Zeichen nicht einlesen werden, wenn der Sonderfall nicht auftritt. CAS RETURN darf ohne ein zwischenzeitliches CAS IN CHAR nicht zweimal hintereinander aufgerufen werden.

Mit CAS TEST EOF kann festgestellt werden, ob ein zeichenweises File, das gerade eingelesen wird, zu Ende ist. Wenn es zu Ende ist, wird bei folgenden Aufrufen der Routine CAS IN CHAR ein Fehler (CY=0) zurückgegeben.

CAS CATALOG liest Block für Block ein und gibt jeweils den Filenamen, den Filetyp und die Nummer des Blocks innerhalb des Files aus. Diese Routine kann nur durch ein Drücken von ESC verlassen werden. Da teilweise andere Meldungen als sonst ausgegeben werden müssen, wird der File-Status besonders gesetzt (Status 5, entspricht CAS CATALOG aktiv).

3.9.4 Die Bearbeitung eines Blocks

Die Routine zum Schreiben eines Blocks auf Band ruft nach der Ausgabe einer Meldung zweimal die Routine CAS WRITE auf, und zwar einmal für den Blockheader und das zweite Mal für den eigentlichen Block. An CAS WRITE wird ein Kenn-Byte übergeben, an dem beim Lesen festgestellt werden kann, ob ein Header oder ein Datenblock folgt. Mit "Datenblock" ist ein 2-KByte-Block gemeint, der Teil eines Programm- oder Datenfiles

sein kann. CAS WRITE ist also eine allgemeine Routine, die einen bestimmten Speicherbereich auf Band schreibt.

Ebenso werden der 64 Byte lange Header und der 2 KByte lange Datenblock jeweils mit CAS READ eingelesen. Weiterhin gibt es noch eine Routine CAS CHECK, die ähnlich CAS READ einen Bereich von Kassette einliest, ihn jedoch nicht abspeichert,sondern mit dem Speicherinhalt vergleicht. Diese Routine wird von Betriebssystem und Basic des CPC nicht aufgerufen. Will man sie für eigene Anwendungen nutzen, so muß man sicherstellen, daß man die richtigen Blocks des richtigen Files mit CAS CHECK vergleicht. Dies erfordert wohl eine Einbindung in übergeordnete Routinen, analog der Einbindung von CAS READ in eine Routine, die die Filenamen vergleicht und auf die richtige Blocknummer achtet.

3.9.5 Das Format eines Blocks

Die auf Kassette zu schreibenden Daten werden seriell über ein Portbit des 8255 ausgegeben und über ein anderes Portbit wieder eingelesen. Jedes Datenbit wird mit zwei Flanken dargestellt. Die Zeiten zwischen den Flanken sind bei einem 1-Bit circa doppelt so lang wie bei einem 0-Bit. Diese Zeiten hängen von der Baudrate ab, die mit CAS SET SPEED gesetzt werden kann. Übergeben werden dieser Routine einmal ein Wert für die Zeiten zwischen zwei Flanken (bei einem 0-Bit) und ein Korrektur-Wert. Man kann hier auch andere als die vom Basic für 1000 oder 2000 Baud benutzten Werte übergeben. Der Korrektur-Zeitwert wird zu der Zeit zwischen zwei Flanken eines 1-Bits addiert und von der Zeit jedes Bits, auf das ein 1-Bit folgt, subtrahiert. Der Korrekturwert soll wahrscheinlich technisch bedingte Veränderungen in den Flankenabständen ausgleichen. Wichtig ist, daß jedes Bit durch zwei gleichlange Zeitabstände zwischen jeweils zwei Flanken dargestellt wird. Das Erzeugen und Messen dieser Zeitabstände geschieht mit Hilfe des Refresh-Zählers.

Jeder Block beginnt mit einer Synchronisationsmarkierung, die aus $0801 1-Bits besteht. Mit Block ist hier ein Bereich gemeint, der sowohl ein Header als auch ein Datenblock ohne Header sein kann. Mit Hilfe dieser Markierung kann beim Lesen die Baudrate ermittelt werden, indem das Mittel über mindestens 256 Zeitabstände zwischen den Flanken gebildet wird. Die Baudrate wird also nicht auf 1000 oder 2000 Baud erkannt, sondern kann auch kleinere, größere oder Zwischenwerte annehmen. Nach der Synchronisationsmarkierung folgt ein einziges 0-Bit. Dieses 0-Bit hat nun halb so große Flankenabstände wie die 1-Bits zuvor. Es wird beim Lesen nach der Baudraten-Erkennung auf zwei solcher kürzeren Flankenzeiten hintereinander gewartet. Je nachdem, ob die zweite Flanke des so erkannten 0-Bits eine High-Low- oder Low-High-Flanke war, ist das Kasset-

tensignal invertiert oder nicht invertiert. Das so gewonnene Invertierungs-Flag wird zum Lesen des ganzen Blocks verwendet.

Nach der Synchronisations-Markierung folgt das Kennbyte des Blocks, das besagt, ob der Block ein Header oder ein Datenblock ist. Danach kommen endlich die eigentlichen Daten des Blocks. Mit "Daten" können hier natürlich auch Bytes aus einem Header gemeint sein. Die Blockdaten werden in Stücken zu 256 Bytes auf das Band geschrieben, nach 256 Bytes folgt ein zwei Byte langes Prüf-Wort, dann wieder 256 Bytes Daten und so weiter. Es kann nun sein, daß die abzuspeichernde Blocklänge kein Vielfaches von 256 ist (ein Header ist z.B. nur 64 Byte lang). In einem solchen Fall werden bis zum Erreichen des nächsten vollen Abschnitts aus 256 Bytes noch Null-Bytes als Füll-Bytes eingefügt. Ein Header besteht demnach also aus 64 Datenbytes und 256 - 64 = 192 Füllbytes.

Abschließend folgt eine der Synchronisationsmarkierung ähnliche Blockende-Markierung, die aus $21 1-Bits besteht.

3.9.6 Die Fehlermeldungen

Die meisten Routinen der unteren Ebene, bis hin zu CAS READ, CAS WRITE und CAS CHECK, zeigen einen aufgetretenen Fehler durch ein gelöschtes Carry an. Im Akku ist dann die Nummer des Fehlers zu finden, wobei eine Null Abbruch durch ESC bedeutet. Beim Schreiben auf Kassette gibt es nur den "Write error a" (Akku = 1). Er zeigt an, daß die Baudrate so hoch ist, daß die Bits nicht mehr schnell genug ausgegeben werden können. Beim Lesen gibt es dagegen vier Fehler: Der "Read error a" (Akku = 1) tritt auf, wenn der Zähler für die Zeit zwischen zwei Flanken überläuft, wenn also die Flanken zu langsam aufeinander folgen. "Read error b" (Akku = 2) zeigt an, daß ein Fehler im Prüf-Wort (Check-Word) gefunden worden ist. CAS CHECK gibt den "Read error c" (Akku=3) zurück, wenn die gelesenen Daten nicht mit den Daten im RAM übereinstimmen. Eine weitere Fehlermöglichkeit, der "Read error d" (Akku = 4), wird in der Routine, die CAS READ aufruft, überprüft. Dieser Fehler wird ausgegeben, wenn ein zu lesender Block länger als 2 Kbyte ist und die Datei zeichenweise gelesen wird, also nur ein Buffer von 2 KByte für den Block zur Verfügung steht. Die übrigen Fehler werden ebenfalls von den CAS READ oder CAS WRITE aufrufenden Routinen ausgegeben.

Auch auf höherer Ebene bestehen Fehlermöglichkeiten. Man kann z.B. versuchen, ein File zweimal zu öffnen, ohne es zwischendurch zu schließen. Solche Fehler werden der aufrufenden Routine mit einem gelöschten Carry gemeldet. Im CPC 464 wird durch CY = 0, Z = 1 und A = 0 ein Abbruch durch ESC angezeigt. Im CPC 664/6128 bedeutet

CY = 0, A = 0 ebenfalls Abbruch, CY = 0, A = $0E zeigt einen Filestatus-Fehler an und CY = 0, A = $0F signalisiert EOF (End of file, Ende des Files).

3.10 Der Editor (EDIT)

Der Editor dient zur Eingabe von Texten in den Computer. Im Gegensatz zu Screen Editoren, die auf einigen Home Computern zu finden sind, handelt es sich beim CPC-Editor um einen Line Editor. Er hat jedoch wegen des Copy Cursors einige Charakteristika von Screen Editoren und stellt tatsächlich einen guten Kompromiß zwischen diesen beiden Kategorien dar.

Der Editor verfügt nur über eine Routine, die von dem Benutzerprogramm angesprungen werden sollte: Edit. Diese Routine holt eine Eingabezeile, die entweder mit einem CR oder mit einem ESC terminiert wird. Diese Zeile wird der aufrufenden Routine nicht auf dem Bildschirm übergeben (darum braucht sich das User-Programm nicht zu kümmern), vielmehr wird der String in einem Buffer abgelegt, dessen Adresse die aufrufende Routine definiert haben muß. Die Länge des Buffers und die laufende Cursorposition innerhalb des Buffers werden auch als Parameter ein- und ausgegeben: die Länge in C, die Position in B. Im ROM-Listing sind dies durchgängige Parameter, daher haben wir sie nicht immer wieder neu aufgeführt.

Bestimmte Tasten haben dabei besondere Funktionen, wie z.B. die Cursor-Tasten und die Copy-Taste. Der Editor realisiert dies derart, daß er sich eine Taste von der Tastatur holt und den ASCII-Code in einer Tabelle sucht. Wenn er ihn findet, so holt er sich die Adresse der Routine, die dieser Taste zugeordnet ist und springt sie an. Die Adresse ist ebenfalls in der Tabelle eingetragen. Findet er den Tastencode nicht, so handelt es sich nicht um eine Funktionstaste, sondern um eine gewöhnliche Taste und er schreibt sie in den Buffer. Entweder schreibt er sie einfach hinein, fügt sie ein oder an. Bei einem CR oder einem ESC kehrt der Editor zurück.

Wir wollen an dieser Stelle die einzelnen Funktionstasten des Editors nicht näher beschreiben. Sie dürften Ihnen aus dem täglichen Gebrauch hinreichend bekannt sein. Auch das ROM-Listing bedarf kaum weiterer Erläuterung, da die Struktur des Editors sehr einfach und linear ist.

Die Editor-Versionen im 464 und im 664/6128 sind ein wenig unterschiedlich, was die Behandlung der übergebenen Zeile angeht. So ist es möglich, dem Editor in dem Buffer, mit dem editiert werden soll, bereits einen String zu übergeben. Dieser String kann beliebig sein, darf jedoch maximal 255 Zeichen umfassen und muß mit einer Null abgeschlossen sein. Im 464 werden die Parameter des Editors, die Bufferlänge und Cursorposition im Buffer, einfach nach dem übergebenen String gesetzt. Im 664/6128 wird überdies noch sichergestellt, daß, für den Fall, daß eine Zahl am Bufferanfang steht, diese durch das Editieren nicht verändert werden kann.

3.11 Das FLOATING POINT PACK (FLO)

3.11.1 Allgemeines

Obschon das FLO-Pack eigentlich nicht mehr zum Operating System zu rechnen ist, haben wir es vorgezogen, es in die Beschreibung des OS mit hineinzunehmen (siehe 3.3).

Das FLO-Pack hat die Aufgabe, sämtliche Arithmetik mit Fließkommazahlen auf dem CPC abzuwickeln, d.h. alle Rechenoperationen mit diesen Zahlen durchzuführen. Die eingebauten arithmetischen Fähigkeiten dieses Packs reichen von einer einfachen Addition über Multiplikation und Division bis hin zum Potenzieren und den wichtigsten irrationalen Funktionen (SIN, EXP, LOG etc.). In diesem Kapitel möchten wir Ihnen einen Einblick verschaffen, wie ein Gerät, das zunächst einmal nur mit ganzzahligen Werten von 0 bis 255 rechnen kann, zu solch erstaunlichen Operationen befähigt ist.

3.11.2 Die Darstellung einer Fließkommazahl

Die erste Frage, die wir stellen müssen, ist: Wie stellt ein 8-Bit Computer Fließkommazahlen dar, die in einen Wertebereich von 1e38 passen?

Wie Sie sich sicher denken können, geschieht die Darstellung solcher Zahlen in einer Art Binärsystem. Um den grundlegenden Gedanken dieser Darstellung besser verstehen zu können, schauen wir uns zunächst einmal die verschiedenen Schreibweisen für uns vertrautere Dezimalzahlen an: so ist da zunächst einmal die "normale" Schreibweise, mit einem Komma zwischen den Stellenwerten 1 und 1/10, z.B. 1234,56. In den Naturwissenschaften, besonders in der Physik, neigt man nun aber dazu, vor allem bei sehr großen bzw. kleinen Beträgen, eine Zahl in der Exponential-Schreibweise anzugeben, unsere Zahl z.B. mit $1.23456 * 10^3$ ($1.23456 * 1000 = 1234.56$). Man versucht also immer genau eine Ziffer ungleich 0 vor dem Komma stehen haben. Vermutlich ist diese Schreibweise Ihnen schon vertraut. Ihr Computer gibt nämlich besonders große oder kleine Beträge in dieser Schreibweise aus: statt 0.000002345 schreibt er $2.345 * 10^{-6}$, bzw. der Kürze halber 2.3456E-6.

Da diese Darstellung neben der Übersichtlichkeit bei größeren Zahlen auch noch einige andere Vorteile bietet, benutzt der Computer sie, um seine Fließkommazahlen darzustellen. Allerdings wählt er nicht das Dezimal-, sondern das Binärsystem. In diesem System gibt es nur die Ziffern 0 und 1. Die Dezimalzahl 2,25 würde binär als 10,01 dargestellt werden. Um dies zu verdeutlichen, gehen wir noch einmal kurz auf den Stellenwert einer Binärziffer ein. Vor dem Komma verdoppelt sich der Stellenwert von Stelle zu

Stelle nach links hin, d.h. direkt vor dem Komma $2^0 = 1$, dann $2^1 = 2$, $2^2 = 4$, $2^3 = 8$ und so weiter. Nach rechts hin nimmt der Zweierexponent immer um eins ab - d.h. in logischer Fortsetzung, daß die Zahl direkt nach dem Komma den Stellenwert $2^{-1} = 1/2$, die nächste Stelle dann $2^{-2} = 1/4$, dann $2^{-3} = 1/8$ etc. hat. Dies läßt sich auch ganz einfach auf unser Dezimalsystem übertragen. Die binäre Zahl 10.01 hat also den Wert $2^1 + 2^{-2} = 2 + 1/4 = 2{,}25$.

Übertragen wir nun die Exponentialdarstellung auf binäre Zahlen. Dazu ein Beispiel: die binäre Zahl 101101,011 (dezimal 45.375) lautet in der (binären) Exponentialdarstellung $1{,}01101011 * 10^{101}$ (der Exponential-Teil ist dezimal 2^5). Diese Darstellung kommt der, die das FLO-Pack für seine Zahlen verwendet, schon recht nahe. Dort werden Fließkommazahlen in einer 32-stelligen Mantisse (das ist der "vordere" Teil der Exponential-Zahl, d.h. der Teil, der vor dem "*" steht) und einem achtstelligen Exponenten dargestellt. Daraus folgt, daß eine Fließkommazahl vier Mantissenbytes (4 * 8 Bit = 32 Bit) und ein Exponentenbyte benötigt. Diese fünf Byte sind im Speicher des CPC wie folgt angeordnet:

```
x+0          x+1   x+2   x+3   x+4
MSB4 (LSB)   MSB3  MSB2  MSB1  (MSB) Exponent
```

In dieser Aufstellung ist x die Adresse der FLO-Zahl (sie muß den einzelnen Routinen des FLO-Packs übergeben werden), MSB bedeutet hier Most Significant Byte (d.h. höchstwertiges Byte), LSB meint Least Signifikant Byte (d.h. niederwertigstes Byte).

Vielleicht ist Ihnen aufgefallen, daß mit dem, was wir bisher gesagt haben, allem Anschein nach der Exponent und auch die ganze Zahl immer positiv sein müssen. Dies ist natürlich nicht der Fall. Wie Sie ja wissen, haben wir die Exponentialdarstellung definiert, indem wir festgelegt haben, daß in der Mantisse immer genau eine Ziffer ungleich Null vor dem Komma stehen muß. Das Komma muß man sich bei obigem Format zwischen b7 und b6 des MSB1 denken, d.h. die eine Vorkommastelle ist das b7 des MSB. Wann immer irgendwelche Operationen mit einer solchen FLO-Zahl durchgeführt wurden, wird sie nachher immer noch einmal normalisiert (korrekt eigentlich "normiert"). Dazu wird die Mantisse so lange verschoben und der Exponent jeweils erhöht oder erniedrigt, bis an dieser Stelle die oberste Eins steht. Da man nun aber bei einer normierten FLO-Zahl davon ausgeht, daß dort eine Eins steht, ist dieses Bit zur Informationsübermittlung eigentlich nicht mehr nötig, es ist redundant. Genau dies nutzt man aus, um nun das Vorzeichen abzuspeichern: Man ersetzt dieses Bit einfach durch das Vorzeichen (0 für +, 1 für -). Wenn mit der Zahl gerechnet werden soll, so wird das Vorzeichen gerettet und die 1 wieder eingesetzt.

Beim Exponenten würde sich z.B. anbieten, die Zweierkomplement-Darstellung zu wählen, um das Problem des fehlenden Vorzeichens zu lösen. Man beschreitet jedoch meist (und so auch im CPC) einen anderen Weg: Zum Exponenten (im Zweierkomplement) addiert man den Wert $81. Dies ergibt dann den FLO-Exponenten. Man macht dies, um den Spezialfall einer Null als FLO-Zahl erfassen zu können. Eine Null kann eigentlich nicht dargestellt werden, da man bei einer FLO-Zahl davon ausgeht, daß das oberste Mantissenbit immer eine Eins ist. Genau dies ist jedoch bei der Null nicht der Fall - hier gibt es in der gesamten Mantisse keine Eins. Man benötigt daher ein besonderes Flag, um eine Null anzuzeigen. Da man dafür nicht extra ein weiteres Byte einführen möchte, nimmt man einen ausgezeichneten Wert des Exponenten dafür. Man hat den Wert $00 dafür gewählt, da dieser besonders leicht abzutesten ist. Die Geschwindigkeit mit der dieser Test durchgeführt werden kann, war also ein sehr wichtiges Kriterium. Da nun jedoch $00 im Exponenten bedeutet, daß der Wert der Zahl Null ist, ist es nicht mehr möglich, die Zweierkomplementsdarstellung für den Exponenten zu wählen. In ihr wird nämlich die Null im Exponenten für die Darstellung der Zahlen von 0,5 <= x < 1 benötigt.

Unsere Zahl $1.01101011 * 10^{101}$ würde demnach im Fließkommaformat wie folgt aussehen: 00 00 80 35 86. (Für Interessierte sei angemerkt, daß man genausogut sagen könnte, zum realen Exponenten würde nur $80 hinzuaddiert. Man muß dann das Komma der normierten Form lediglich noch um eine Stelle nach rechts verschieben, d.h. die Mantisse hat vor dem Komma nur noch Nullen, die höchste Eins folgt gleich nach dem Komma. Beide Darstellungen des Sachverhaltes sind gleichwertig.)

3.11.3 Beispiel einer Operation: die Addition

Nun, da Sie wissen, wie FLO-Zahlen abgespeichert werden, möchten Sie sicher gerne wissen, wie damit gerechnet wird. Wir können hier nicht auf alle Grundrechenarten eingehen. Allerdings halten wir es für sinnvoll, wenigstens ein einfaches Beispiel zu geben. Wir wollen uns daher einmal anschauen, wie zwei FLO-Zahlen addiert werden.

Zunächst einmal wird nachgeschaut, ob die Vorzeichen identisch sind. Sind sie gleich, so werden die Beträge addiert, sind sie ungleich, subtrahiert. Tatsächlich ist die Subtraktion nichts anderes als das Invertieren des Vorzeichens des entsprechenden Operanden und anschließender Addition.

Wenn die Beträge addiert werden, so werden zunächst die Exponenten angeglichen. Dazu werden die beiden Exponenten verglichen, um festzustellen, welcher der größere ist. Der kleinere wird dann auf den Wert des größeren gebracht, indem die Mantisse nach rechts verschoben wird (bzw.

das Komma nach links) und der Exponent für jede Bitposition, um die verschoben wurde, um eins erhöht wird. In der Mantisse wurde natürlich, wie eigentlich vor jeder Operation, das Vorzeichenbit vorher durch die obligatorische Eins ersetzt.

Nachdem beide Exponenten angeglichen sind, werden die beiden Mantissen wie ganz normale 32-Bit-Integer byteweise addiert. Sollte ein Übertrag über das höchste Bit auftreten, so wird die Mantisse noch einmal nach rechts verschoben und der Exponent erhöht. Zum Schluß wird dann das Vorzeichen an seine Stelle statt des dortigen 1-Bits gesetzt.

Wenn die Vorzeichen ungleich sind, so ist das Verfahren ganz ähnlich: auch hier werden zunächst die Exponenten angeglichen. Dann jedoch werden die Mantissen subtrahiert. Ist die zweite Mantisse größer als die erste, so tritt ein Borger von der höchsten Mantissenstelle auf. Dann wird das Vorzeichen des ersten Operanden invertiert (dies ist dann das Vorzeichen des Ergebnisses) und das Zweierkomplement der Mantisse wird gebildet. Wenn das höchste Bit gesetzt ist, wird nur noch gerundet und das errechnete Vorzeichen eingesetzt. Andernfalls muß das Ergebnis noch normiert werden.

3.11.4 Die irrationalen Funktionen

Der Laie fragt sich oft, wie der Computer wohl Funktionen wie SIN, COS oder EXP berechnet. Viele vermuten, daß der Computer über eine Tabelle verfügt, aus der er die Funktionswerte für die einzelnen Argumente herausliest. Bei näherer Betrachtung jedoch erweist sich der Gedanke als nicht realisierbar, da eine solche Tabelle eine gigantische Länge haben müßte. Die Verfahren, die digitale Rechner heute zur Ermittlung von Funktionswerten solcher Funktionen benutzen, sind vielmehr mathematischer Natur. Es handelt sich dabei um "Reihenentwicklungen", d.h. um theoretisch unendlich lange Ausdrücke, die aus unendlich vielen Gliedern bestehen. Der Computer verwendet zur Berechnung eines Funktionswertes nur die ersten Glieder einer solchen Reihe, die ihm bereits eine gute Näherung bringen. (Es gibt auch endliche Reihen, allerdings nicht für die Funktionen, die wir hier betrachten.)

Eine solche Reihe für allgemeine, unendlich oft differenzierbare Funktionen f ist die "TAYLORsche Reihe". Sie hat die folgende Form:

$$(1) \qquad f(x) = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \ldots + \frac{(x-a)^n}{n!} f^{(n)}(a)$$

Hierin ist a ein fester Wert aus dem Definitionsbereich der Funktionen f, f', f''... $f^{(n)}$. Sinnvollerweise wählt man für "a" einen ausgezeichneten Wert, bei dem möglichst viele Glieder fortfallen bzw. sich vereinfachen lassen. Oft ist es z.B. sinnvoll, "a" gleich null zu setzen. Damit ergibt sich dann ein Spezialfall der Taylor'schen Reihe, die "MacLAURINsche Reihe". Sie hat die Form

$$(2) \qquad f(x) = f(o) + \frac{f'(o)}{1!}x + \frac{f''(o)}{2!}x^2 + \ldots + \frac{f^{(n)}(o)}{n!}x^n$$

Da der Computer nur eine begrenzte Anzahl von Koeffizienten zur Berechnung einer Reihe heranziehen kann, folgt, daß die Ergebnisse immer ein wenig ungenau sind. In der Mathematik begegnet man diesem Effekt, indem man ein "Restglied" näherungsweise bestimmt. Bei der Berechnung eines Funktionswertes im Computer versucht man dagegen die Ungenauigkeiten zu minimieren, indem man die Koeffizienten, die man zur Berechnung benutzt, ein wenig modifizert, das Restglied praktisch noch mit einfließen läßt. Die Genauigkeit der benutzten Näherungen wird zudem auch noch dadurch gesteigert, daß man das Argument normiert. Es wird in einen bestimmten Zahlenbereich "gezwungen", in dem die verwandte Näherung mit besonders hoher Genauigkeit gilt. Das Ergebnis aus dieser Näherung wird dann in entsprechender Weise wieder so umgeformt, daß die Umformungen berücksichtigt werden, die zur Normierung nötig waren.

Die Näherung selber geschieht mit einer "Polynomberechnung", d.h. es wird ein Polynom P(x) der Form

$$(3a) \qquad P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n$$

berechnet. Wird x zuvor quadriert, so hat das Polynom die folgende Form:

$$(3b) \qquad P(x) = a_0 + a_1x^2 + a_2x^4 + a_3x^6 + \ldots + a_nx^{2n}$$

Die Koeffizienten a_1 bis a_n sind in einer Tabelle abgelegt, die direkt hinter dem Aufruf der Polynom-Routine steht, wobei a_n als erstes dasteht, bis hinunter zu a_0 als letztem Koeffizient. Je nach Näherung wird manchmal auch noch das errechnete Polynom mit dem Eingangsargument (dem x) multipliziert, d.h. es ergibt sich ein Polynom P(x) der Form

$$(4a) \qquad P(x) = a_0 + a_1x^2 + a_2x^3 + a_3x^4 + \ldots + a_nx^{n+1}$$

bzw. mit zuvor quadriertem Argument die Form

$$(4b) \qquad P(x) = a_0 + a_1x^2 + a_2x^5 + a_3x^7 + \ldots + a_nx^{2n+1}$$

Schauen wir uns hier nun die Näherungen für die einzelnen Funktionen an. Wir haben darauf verzichtet, jede einzelne entwickelte Reihe herzuleiten. wir begnügen uns im Rahmen dieses Buches damit, sie einfach nur aufzuführen. Dem interessierten Leser ist es sicherlich möglich, sich die Reihen selber zu entwickeln.

3.11.4.1 Die Reihe der LOG-Funktion

Die Reihe für die LOG-Funktion wurde aus der Reihe für die artanh-Funktion (areatangens hyperbolicus) nach der Beziehung

$$(5) \qquad \ln \frac{1+x}{1-x} = 2 \operatorname{artanh} x$$

entwickelt. Daraus läßt sich dann - durch Entwicklung der Taylor'schen Reihe für den artanh - die im CPC verwandte Näherung für den ln (logarithmus naturalis) herleiten:

$$(6) \qquad \ln x = 2\left[y + \frac{y^3}{3} + \frac{y^5}{5} + \ldots\right]; \; y = \frac{x-1}{x+1}$$

Das Argument wird zunächst entsprechend normiert und in den Bereich zwischen SQR(1/2) und SQR(2) gebracht. Dann wird das Polynom berechnet (vorher wird natürlich das Eingangsargument y aus dem sich ergebenden x berechnet), und zwar nach der Formel 4b. Das Ergebnis dieser Polynomberechnung ist jedoch, wie an den Koeffizienten unschwer zu erkennen ist, durch ln2 geteilt. Dies wird ausgeglichen, indem das Ergebnis später mit lb Basis (d.h. LOG(Basis)/LOG(2)) multipliziert wird. Vorher wird jedoch noch der Binärexponent der Zahl addiert (dieser wurde vorher beim Normieren der Zahl gerettet).

3.11.4.2 Die Reihen der EXP-Funktion

Bei der EXP-Funktion war es uns leider nicht möglich, einen sinnvollen Zusammenhang herzustellen. Dies lag vor allem daran, daß zwei Polynome P1(x) und P2(x) berechnet und miteinander verknüpft werden. Wir können daher nur die uns zur Verfügung stehenden Erkenntnisse reproduzieren.

Exp(x) wird demnach wie folgt berechnet: Zunächst wird x/ln2 gebildet. Dies geschieht, um 2 nachher mit dem ganzzahligen Teil dieses Ausdrucks nach der Formel $2^{(x/\ln 2)} = 2^{(x*\mathrm{lbe})} = (2^{\mathrm{lbe}})^x = e^x = \exp(x)$. Diesen ganzzahli-

gen Teil von x/ln2 nennen wir Y. Es gilt also Y = int(x/ln2). Durch die obige Umformung muß nur noch der Nachkommateil von x/ln2 genähert werden. Wir nennen diesen Teil y. Es gilt damit y = x/ln2-Y. Diese Beschränkung erhöht natürlich die Genauigkeit. Es werden dann zwei Polynome $P1(y^2)$ und $P2(y^2)$ nach uns, wie gesagt, unbekannten Reihen berechnet. Die volle Formel lautet dann:

$$\text{(7)} \qquad \exp x = 2^{Y+1}\left[\frac{P_2(y^2)}{P_1(y^2) - P_2(y^2)} + 0.5\right];\ Y = \operatorname{int}\left(\frac{x}{\ln 2}\right);\ y = \frac{x}{\ln 2} - Y$$

Sollte es Ihnen gelingen, die Zusammenhänge zu ergründen, so wären wir sehr erfreut, wenn Sie sie uns wissen ließen.

3.11.4.3 Die Reihe der SIN/COS-Funktionen

Diese Reihe ist nach der MacLaurin'schen Formel entwickelt worden. Um sie anwenden zu können, wird x zunächst einmal durch Pi (bzw. 180 für DEG) geteilt. Um die Phasenverschiebung auszugleichen wird bei COS 0,5 addiert. Ist der ganzzahlige Teil von x/Pi (bzw. x/Pi + 0,5 bei COS) ungerade, so befindet sich die Funktion in der negativen Halbperiode, d.h. das Vorzeichen des Ergebnisses muß invertiert werden. Im folgenden ist der ganzzahlige Teil ohne Bedeutung. Der Nachkommateil wird noch mit zwei multipliziert. Er ist also insgesamt mit 2/Pi multipliziert worden. Der Faktor 2/Pi wird durch die Koeffizienten ausgeglichen.

Die Reihenentwicklung für die SIN-Funktion benutzt einige besondere Eingenschaften der trigonometrischen Funktionen. So ist f(0) (das ist hier SIN(0)) und alle geraden Ableitungen immer gleich null. Dann gilt, daß sin'x = cos x, sin"x = -sin x (daher sind alle geraden Ableitungen null), sin'"x = -cos x und schließlich sin""x = sin x und so weiter. Daraus folgt, daß nur jedes zweite Glied der MacLaurin'schen Reihe ungleich null ist, und daß die ungeraden Ableitungen von sin x nur entweder +1 (cos 0) oder -1 (-cos 0) sein können. Dies zeigt sich in den alternierenden Vorzeichen der Koeffizienten. Auch in den Koeffizienten ist ein ausgleichender Faktor für die Multiplikation mit 2/Pi enthalten. Die Formel, die der Näherung damit schließlich zugrunde liegt, lautet demnach:

$$\text{(8)} \qquad \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + - \ldots$$

3.11.4.4 Die Reihe der ATN-Funktion

Die Reihe der ATN-Funktion ist, wie man zeigen kann, eine MacLaurin'sche Reihe, mit arctan(0) = 0. Sie lautet wie folgt:

(9) $$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + - \ldots$$

3.12 Das INTEGER PACK (INT)

Das Integer Pack verwirklicht die Integer-Arithmetik im CPC. Es kann sämtliche Grundrechenarten mit 16-Bit Zweierkomplement-Zahlen durchführen. Dies schließt auch die Division ein, die dann natürlich zwei Zahlen als Ergebnis liefert: den Quotienten und den ganzzahligen Rest. Der Operator DIV (im CPC-Basic das \-Zeichen) gibt den Quotienten, der Operator MOD den Rest als Ergebnis zurück.

Da die Integer-Arithmetik relativ einfach ist, verzichten wir darauf, an dieser Stelle näher auf sie einzugehen. Es sei nur noch erwähnt, daß das Integer Pack ein ganz besonderes Schicksal erleiden mußte: im Zuge der Erweiterungen des Betriebssystems im CPC 664/6128 wurde es aus Platzgründen einfach aus dem unteren ROM entfernt und in das obere ROM gesteckt. Hieran erkennt man, daß es nicht denselben Status eines Packs hat, wie z.B. das Kernel oder der Keyboard Manager. Auch das FLO-Pack und der Editor sind solche Stiefkinder im Operating System.

4 Das BASIC des CPC

4.1 Speicherorganisation

4.1.1 Aufteilung in ROM und RAM

Zur Erinnerung zunächst einmal die grobe Speicheraufteilung des CPC in ROM und RAM: Der CPC besitzt 64 KByte bzw. 128 KByte RAM und 32 KByte ROM, die durch Banking mit einem Adreßraum von nur 64KByte verwaltet werden können. Zusätzlich sind noch bis zu 251 externe ROMs anschließbar.

Das Basic benutzt auch im CPC 6128 nur 64 KByte der 128 KByte RAM. Die vollen 128 KByte können über eine RSX-Erweiterung angesprochen werden, die von Diskette geladen werden muß. Es sind auch Experimente mit der Routine KL RAM SELECT oder direkt mit dem Gate Array denkbar (siehe Abschnitte 1.3.7 und 3.1.3).

4.1.2 Die Aufteilung des RAMs

Die 64 KByte RAM des CPC 464/664 bzw. Bank 0 bis 3 des CPC 6128 sind zunächst zwischen Basic und Betriebssystem aufgeteilt. Das Betriebssystem beansprucht hierbei die Bereiche $0000-$003F und $B100-$FFFF, das Basic hingegen $0040-$B0FF. Eine genauere Aufteilung dieser Bereiche finden Sie in Abbildung 4.1 auf der nächsten Seite.

4.1.3 Der Basic-Anwenderbereich

Der Basic-Anwenderbereich ist noch einmal unterteilt. Die einzelnen Adressen liegen hier jedoch nicht fest, da beispielsweise die Programmlänge ja variieren kann. Deshalb gibt es im Systembereich des Basic mehrere Zeiger, die den Beginn eines neuen Bereichs markieren. Vom Betriebssystem werden dem Basic nach einem Kaltstart zwei Zeiger übergeben: Lo-RAM (bei $0040) und Hi-RAM (bei $ABFF). Der durch diese Zeiger eingegrenzte Bereich kann durch Erweiterungen in externen ROMs noch verkleinert werden. Basic ruft die Routine KL ROM WALK auf, mit der festgestellt wird, ob externe ROM-Erweiterungen des Basic, genannt Resident System Extension (RSX), vorhanden sind. Jedes ROM reserviert sich bei KL ROM WALK seinen eigenen Speicherbereich durch Heruntersetzen von Hi-RAM. Den endgültigen Hi-RAM-Zeiger speichert Basic dann in einer Systemvariablen.

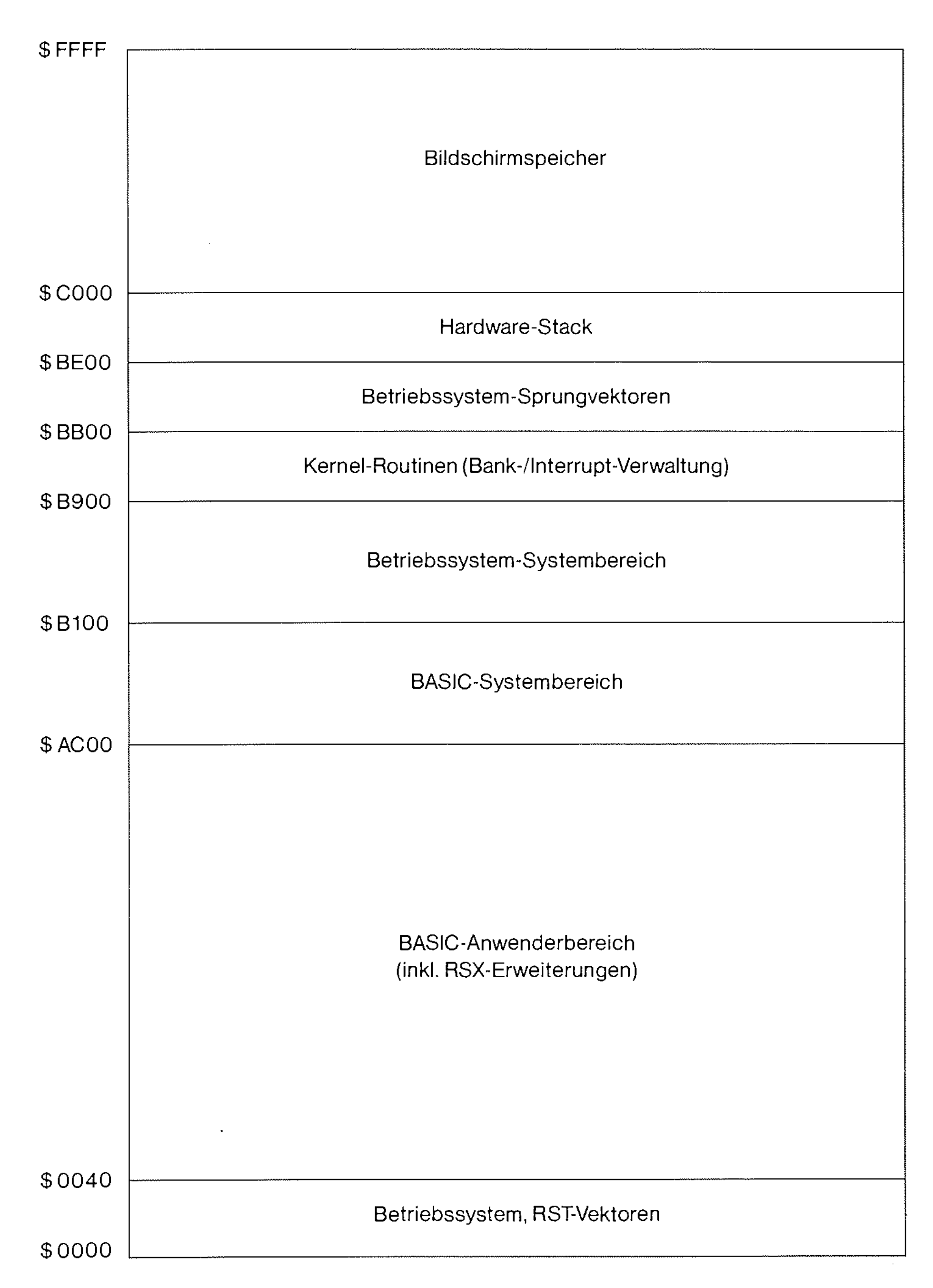


Abbildung 4.1: *Die Aufteilung des RAM*

Es folgt nun eine Tabelle der RAM-Zeiger des Basic, die die Aufteilung des Bereichs zwischen Lo-RAM und Hi-RAM festlegen. Nicht alle dieser Zeiger geben den Start eines Bereichs an, einige zeigen auch auf das Ende des vorhergehenden Bereichs, das eine Adresse tiefer liegt. Diese Zeiger sind in der Tabelle mit einem Stern (*) gekennzeichnet.

464	664/6128	
($AE7F)	($AE62)	LoRAM
		Token-Buffer
($AE81)	($AE64)	Basic-Programmstart
		Basic-Programm
($AE83)	($AE66)	Basic-Programmende
		(frei)
($AE85)	($AE68)	Variablenstart
		einfache Variablen
($AE87)	($AE6A)	Start der Felder
		Feldvariablen
($AE89)	($AE6C)	Ende der Felder
		frei
($B08D)	($B071)	* Start der Strings
		Inhalte der Stringvariablen
($B08F)	($B073)	* Ende der Strings
		(frei)
($AE7B)	($AE5E)	* HIMEM
		frei für Maschinenprogramme u.a.
($AE7D)	($AE60)	* HiRAM

In der Tabelle tauchen zwei mit "(frei)" gekennzeichnete Bereiche auf, die zwar nicht belegt sind, jedoch normalerweise keine Ausdehnung haben, da der Start des jeweiligen Bereichs mit dem Start des nächsten Bereichs zusammenfällt. Es exisitieren trotzdem zwei verschiedene Zeiger, da in einigen Routinen des Basic die Zeiger vorübergehend unterschiedliche Werte besitzen können. So wird beim Mergen eines Programms von Kassette oder Diskette der Variablenbereich geschützt, damit er trotz der Programmänderungen erhalten bleibt. Dazu wird der Variablenbereich direkt unter den Stringbereich geschoben, so daß der freie Basic-Bereich jetzt zwischen Programmende und Variablenstart liegt. Der CPC 464 setzt den Start der Strings dann auf den neuen Variablenstart, um die Variablen zu schützen. Dagegen liegt beim CPC 664 und beim CPC 6128 zwischen den Zeigern auf Programmende und Variablenstart tatsächlich ein freier Bereich. Nach Ausführung der Merge-Routine wird der Variablenbereich dann wieder ans Programmende geschoben.

4.1.4 User-Matrizen und Ein-/Ausgabebuffer

Die Definition von Zeichenmatrizen durch den Benutzer sowie die Speicherung von Daten auf Kassette oder Diskette erfordern jeweils einen bestimmten Speicherbereich, dessen Adresse dem Text-Pack bzw. dem

Cassette-Manager übergeben werden muß. Im Falle der User-Matrizen ist die Größe des benötigten Bereichs in Byte gleich der Zahl der umdefinierten Zeichen mal acht, während die Größe des Ein- und des Ausgabebuffers jeweils 2 KByte beträgt, zusammen also 4 KByte. Ein-/ Ausgabebuffer werden immer gemeinsam als 4-KByte-Block reserviert, auch wenn nur einer von ihnen tatsächlich benutzt wird.

Für die Ein-/Ausgabebuffer und die User-Matrizen wird immer direkt unterhalb HIMEM Platz reserviert, nachdem der Stringbereich entsprechend nach unten verschoben wurde. Anschließend wird HIMEM unterhalb des Starts des reservierten Bereichs gesetzt. Beim CPC 464 wird zusätzlich das Ende des freien RAMs (HiRAM) auf den neuen Wert von HIMEM gesetzt und das alte HiRAM für den Fall der Freigabe des reservierten Bereichs zwischengespeichert. Beim CPC 664/6128 wird dagegen HIMEM zwischengespeichert, während HiRAM erhalten bleibt.

4.2 Der Basic-Compreter

Im Handbuch des CPC ist von einem Basic-Interpreter die Rede, der die Basic-Programme ausführt. In der Tat wird der Text nicht - wie bei vielen anderen Programmiersprachen - in einem Stück übersetzt (compiliert) und dann der compilierte Code ausgeführt, sondern der Text wird während der Ausführung interpretiert. Ein Interpreter hat den Vorteil, daß er interaktiv arbeitet, der Benutzer also das Programm ändern und die Wirkung sofort beobachten kann, ohne zwischendurch zeitaufwendig compilieren zu müssen. Die eigentliche Ausführung dagegen ist bei einem Interpreter meist langsamer als bei einem Compiler. Bei Programmiersprachen wie z.B. Pascal oder C kann ein einzelner Programmteil aufgrund bestimmter Daten- und Programmstrukturen nicht unabhängig vom Rest des Programms ausgeführt werden. Diese Sprachen eignen sich daher nicht so gut als Interpretersprache. Basic hat dagegen nur wenige Programmstrukturen und erlaubt keine komfortable Realisierung von Datenstrukturen und kann deshalb mit relativ geringem Aufwand als Interpretersprache betrieben werden.

Das Scheider-Basic geht einen Mittelweg zwischen Interpreter und Compiler. Jede einzelne Programmzeile wird schon in gewissem Grade übersetzt, bevor sie ins Programm eingefügt wird. Die Zeile wird bei der Ausführung aber dennoch interpretiert. Die Interpretation geht aber schneller vonstatten als bei einer nicht übersetzten Zeile. Eine solche Interpreterart wird auch "Compreter" genannt.

4.3 Die Speicherung des Basic-Programms

4.3.1 Die Tokenisierung einer Zeile

Beim Vor-Übersetzen einer Programmzeile wird jedes Basic-Schlüsselwort (keyword) wie z.B. "LIST", "SIN" oder auch "*" und "<=" durch ein kürzeres Zeichen (Token) ersetzt. Die tokenisierte Zeile ist nicht nur schneller in der Ausführung, sondern spart auch Speicherplatz. Es werden nicht nur die Basic-Keywords tokenisiert, auch Zahlenkonstanten und Variablen werden mit einem Token versehen. Zahlenkonstanten werden von ihrer ASCII-Darstellung in das binäre Speicherformat des Rechners gebracht.

RSX-Befehlsworte werden nicht komprimiert, sondern in ihrer ASCII-Darstellung zusammen mit dem RSX-Kennzeichen gespeichert. Ebenso wird mit Variablennamen verfahren. Einige Besonderheiten bei der Tokenisierung werden später noch erklärt.

4.3.2 Die Speicherung der Zeilen im Basic-Programm

Die Zeilen des Basic-Programms werden in der Reihenfolge ihrer Zeilennummern abgespeichert. Jede Zeile benötigt zur Einbindung ins Programm noch fünf Verwaltungsbytes. Die Zeile beginnt mit einem zwei Byte langen Offset zur nächsten Zeile, also mit der Länge der Zeile. Mit diesem Offset kann das Programm schnell durchsucht werden. Nach dem Offset bzw. der Länge folgt die Zeilennummer, ebenfalls in zwei Bytes. Nach dem sich anschließenden tokenisierten Zeilen-Text wird als Endmarkierung eine Null gespeichert. Am Anfang des Basic-Programms steht ebenfalls eine Null. Jede Zeile wird also von zwei Nullen eingeschlossen. Nach der Null am Zeilenende der letzten Zeile des Programms folgt ein Offset Null (also zwei Nullbytes). Hierdurch wird das Programmende markiert.

4.3.3 Die Speicherung von Zeilennummern

Zeilennummern im Programmtext (nach GOTO, GOSUB usw.) werden von der ASCII-Darstellung in die binäre Darstellung gebracht und mit einem Token versehen. Die Zeilennummern werden während der Programmausführung durch Zeilenadressen ersetzt. Zeilenadressen haben den Vorteil, daß die zugehörige Zeile nicht im Programm gesucht werden muß. Die Zeilenadressen erhalten ein eigenes Token. Es wird genau genommen die Adresse der Null vom Zeilenende der vorigen Zeile gespeichert.

Beim RENUM-Befehl müssen nicht nur die Zeilennummern vor jeder Zeile, sondern auch die Zeilennummern im Programmtext umnumeriert werden. Um den Aufwand gering zu halten, werden letztere vor der

Umnumerierung der Zeilen durch Zeilenadressen ersetzt. Es brauchen dann nur die Zeilennummern vor jeder Zeile neu numeriert zu werden.

Zeilenadressen im Programm haben auch einen Nachteil: Das Programm ist nicht mehr ortsunabhängig (position independent). Dies ist jedoch notwendig, wenn Teile des Programms verschoben werden müssen, um eine neue Zeile einzufügen, oder wenn das Programm abgespeichert werden soll. Die Zeilenadressen im Programm werden in solchen Fällen wieder durch Zeilennummern ersetzt. Die Systemvariable $AE3A ($AE21 beim CPC 664/6128) ist gleich null, wenn das Programm ortsunabhängig ist.

4.3.4 Die Tokenisierung von Variablennamen

Das Schneider-Basic erlaubt drei verschiedene Variablentypen: Integervariablen (z.B. A%), Stringvariablen (z.B. A$) und REAL-Variablen (z.B. A!). Die Tokens für mit "%", "$" bzw. "!" gekennzeichnete Variablennamen lauten $02, $03 bzw. $05. Es besteht die Möglichkeit, auch unmarkierte Variablen zu benutzen, deren Typen dann entsprechend der DEFINT, DEFSTR oder DEFREAL-Definition für den Anfangsbuchstaben der jeweiligen Variablen gesetzt werden.

Der Name einer Variablen wird in seiner ASCII-Darstellung abgelegt, das höchste Bit des letzten Namensbytes ist als Endmarkierung gesetzt. Der Name darf maximal 40 Zeichen lang sein. Zwischen Token und Namen wird noch ein zwei Byte langer Offset eingefügt, dessen Bedeutung später erklärt wird. Dieser Offset wird bei der Tokenisierung einer Zeile zunächst als Kennzeichen dafür, daß er nicht gültig ist, auf Null gesetzt. Stößt der Basic-Interpreter bei der Programmausführung auf ein Variablennamen-Token, so wird der zugehörige Offset eingesetzt, um den Variableneintrag schneller auffinden zu können.

Ein unmarkierter Variablenname (ohne "%", "$" oder "!") wird bei der Tokenisierung durch das Token $0D kenntlich gemacht. Wenn der zugehörige Offset bei Ausführung des Programms eingetragen wird, stellt der Interpreter den Typ der Variablen (entsprechend DEFINT, DEFSTR und DEFREAL) fest und setzt das Token bei einer Integervariablen auf $0B, bei einer Stringvariablen auf $0C und bei einer REAL-Variablen auf $0D. Das Token $0D hat also zwei Bedeutungen: bei nicht eingetragenem Offset (Offset = 0) bedeutet es, daß eine unmarkierte Variable vorliegt, deren Typ noch nicht festgestellt ist. Andernfalls handelt es sich um eine unmarkierte REAL-Variable.

4.4 Die Speicherung von Variablen

4.4.1 Speicherung von numerischen Werten

Integerwerte werden als Zwei-Byte-Werte im Zweierkomplement gespeichert, REAL-Werte als aus Vorzeichen, Mantisse und Exponent bestehende 5-Byte-Werte. Das genaue Format wurde in Kapitel 3.11 beschrieben.

4.4.2 Speicherung von Strings

Strings haben im Schneider-Basic keine feste Länge. Für eine Stringvariable kann im Variablenbereich aber nur ein fester Platz reserviert werden. Bei Strings wird deshalb im Variableneintrag nur ein sogenannter Stringdescriptor gespeichert. Ein Stringdescriptor besteht aus Länge und Adresse des Strings, also aus drei Bytes. Falls die Länge null ist, hat die Adresse einen beliebigen (undefinierten) Wert. Der eigentliche String wird in einem eigens dafür vorgesehenen Bereich gespeichert, der durch zwei Basic-Systemzeiger eingegrenzt wird. Während beim CPC 464 lediglich der String Zeichen für Zeichen abgelegt wird, reservieren CPC 664 und CPC 6128 vor jedem String noch zwei Bytes, in denen normalerweise die Länge des Strings abgelegt wird. Für die Stringverarbeitung spielen diese beiden Bytes keine weitere Rolle, benötigt werden sie nur bei der Garbage Collection.

4.4.3 Die Garbage Collection

Wenn eine Stringvariable einen neuen Wert zugewiesen bekommt, so muß für den neuen String neuer Platz im Stringbereich reserviert werden. Der Platz für den alten String bleibt also ungenutzt erhalten. Auf diese Weise können sich eine Reihe ungültiger Strings im Stringbereich ansammeln. Wenn kein Speicherplatz für weitere Strings mehr da ist oder anderweitig mehr Platz als vorhanden benötigt wird, so müssen die ungültigen Strings beseitigt werden. Diesen Vorgang nennt man Garbage Collection ("Abfallsammeln").

Im CPC 464 wird die Garbage Collection wie folgt durchgeführt. Zunächst wird der Zeiger auf den Start des Stringbereichs und dort auf den Wert des Stringbereich-Endzeigers gebracht. Es werden also quasi alle Strings aus dem Stringbereich gelöscht. Dann werden alle Stringdescriptoren durchgesehen und der Descriptor mit der höchsten Stringadresse herausgesucht. Der zugehörige String wird dann direkt unter den neuen Stringbereich geschoben und in ihn aufgenommen. Dann wird wiederum der String mit der höchsten Adresse außerhalb des Stringbereichs gesucht und verschoben, bis alle Strings wieder im Stringbereich sind. Da nur die Strings, zu denen ein Descriptor existiert, in den Stringbereich aufgenommen wurden, sind

jetzt keine ungültigen Strings mehr im Stringbereich. Der zur Verfügung stehende freie Platz ist eventuell größer geworden.

Bei der Garbage Collection des CPC 464 werden für jeden String sämtliche Descriptoren durchgegangen und überprüft, um die höchste Stringadresse außerhalb des Stringbereichs zu finden. Bei n Descriptoren erfordert dies n^2 Schritte. Es dauert bei Benutzung von großen Stringfeldern also seine Zeit, bis die Garbage Collection fertig ist. Deswegen wurde im CPC 664 und im CPC 6128 ein schnelleres Verfahren gewählt. Sämtliche Descriptoradressen werden in die zwei freien Bytes vor dem zugehörigen String geschrieben. Die Länge, die vorher in diesen Bytes stand, wird in den Descriptor eingetragen (obwohl sie dort schon enthalten ist). Dann werden sämtliche Strings durchgegangen und nach unten verschoben. Die Strings ohne Descriptor, bei denen also keine Descriptoradresse eingetragen ist, werden nicht berücksichtigt. Anschließend wird der gesamte Stringbereich soweit wie möglich nach oben verschoben und dann die Stringadressen in den Descriptoren aktualisiert. Da zu einem String sofort der Descriptor festgestellt werden kann, brauchen bei diesen Schritten also nicht jedesmal alle Descriptoren für jeden String untersucht werden. Die benötigte Zeit ist nicht mehr quadratisch, sondern nur linear abhängig von der Zahl der Descriptoren.

4.4.4 Die Speicherung einfacher Variablen

Einfache Variablen, einfach im Unterschied zu Feldvariablen, werden in einem durch zwei Systemzeiger eingegrenzten Bereich gespeichert. Die Variablen werden in diesen Bereich in der Reihenfolge ihres Auftretens bei der Programmausführung eingetragen. Der Eintrag einer Variablen ist wie folgt aufgebaut:

Offset der nächsten Variablen in der VL (2 Bytes)
Name der Variablen (1 bis 40 Bytes)
Typ der Variablen (1 Byte)
1 = Integer
2 = String
4 = REAL
Variablenwert bzw. Stringdescriptor (2, 3 oder 5 Bytes)

Um den zugehörigen Variableneintrag zu einem Variablennamen schnell finden zu können, existieren 26 verschiedene verkettete Listen (VL) von einfachen Variablen, für jeden möglichen Anfangsbuchstaben eine. Die ersten beiden Bytes eines Variableneintrags dienen zum Einhängen der Variablen in die entsprechende verkettete Liste. In diesen Bytes wird jedoch nicht die Adresse der nächsten Variablen in der Liste, sondern ein Offset zur nächsten Listenvariablen relativ zum Start der einfachen Variablen gespeichert.

Ein ebensolcher Offset wird auch nach einem Variablen-Token in das Programm gespeichert, wenn der Interpreter auf ein solches Token stößt. Der Variableneintrag kann dann einfach gefunden werden: zu dem Offset wird der Start der einfachen Variablen addiert. Dieser Wert muß jedoch noch um eins korrigiert werden, da der um eins erhöhte Offset abgespeichert wird. Dies ist notwendig, um den Offset null, der ja auftreten kann, von einer Null zu unterscheiden, die bedeutet, daß noch kein Offset eingetragen wurde bzw. die verkettete Variablen-Liste zu Ende ist.

Die Speicherung von Offsets anstelle von absoluten Adressen hat den Vorteil, daß der Variablenbereich ortsunabhängig ist, also beliebig verschoben werden kann. Verschiebungen des Variablenbereichs sind nötig, wenn eine Programmzeile eingefügt oder gelöscht werden soll.

4.4.5 Die Speicherung von Feldvariablen

Feldvariablen werden vom Schneider Basic in einem gesonderten Bereich gespeichert. Jeder Feldvariablen-Eintrag beginnt mit einem Kopf, der wie folgt aussieht:

Offset des nächsten Feldes in der VL (2 Bytes)
Name des Feldes (1 bis 40 Bytes)
Typ des Feldes (1 Byte)
    1 = Integer
    2 = String
    4 = REAL
Länge des Feldes ab dem Dimensionsbyte (2 Bytes)
Zahl der Dimensionen des Feldes (1 Byte)
Tabelle der maximalen Indizes (dim * 2 Bytes)

Die Tabelle der maximalen Indizes enthält die beim DIM-Befehl übergebenen Indizes. Nach dieser Tabelle folgen die eigentlichen Feldelemente als 2-Byte-Integer, 5-Byte-REAL-Werte oder 3-Byte-Stringdescriptoren.

Die Felder sind nicht nach dem Anfangsbuchstaben des Namens, sondern entsprechend ihres Typs in eine von drei verketteten Listen eingehängt. Der Kettungs-Offset bezieht sich nicht auf den Start der Variablen, sondern auf den Start der Felder. Auch bei den Feldern wird der um eins erhöhte Offset abgespeichert. Offsets im Programm beziehen sich auf den Start des Feldes, die Adresse des gesuchten Elements muß also in jedem Fall anhand der aktuellen Indizes berechnet werden.

4.5 Die Auswertung des tokenisierten Basic-Textes

4.5.1 Eingabe- und Interpreterschleife

Der Basic-Interpreter besteht auf höchster Ebene aus zwei Schleifen: aus der Eingabe- und der Interpreterschleife. Die Eingabeschleife holt mit Hilfe des Editors eine Eingabezeile und wertet diese aus. Die Zeile wird tokenisiert und gegebenenfalls ins Programm eingefügt. Falls es sich um eine Direkteingabe handelt, wird die Interpreterschleife angesprungen.

Die Interpreterschleife führt sowohl Direkteingaben als auch Programm aus. Bei einem Abbruch durch ESC, bei Programmende und bei den Befehlen END und STOP wird wieder die Eingabeschleife angesprungen. Da nach einer tokenisierten Direkteingabe drei Nullen folgen müssen, wird das Ende der Direkteingabe als "Programmende" erkannt. Im Programm folgen nach der Null am Ende der letzten Zeile zwei weitere Nullen als Programmende-Kennzeichen.

Die Interpreterschleife holt das nächste Token und überprüft, ob es zu einem Befehl gehört. Wenn nicht, wird ein Fehler ausgegeben oder der LET-Befehl angesprungen, da das LET-Token wahlfrei ist. Andernfalls wird die Befehlsadresse entsprechend des Tokens aus einer Tabelle geholt und eine Routine für den Befehl angesprungen. Die Interpreterschleife kümmert sich um Zeilen- und Programmende, um ein eventuelles Tracing und um Programmunterbrechungen und -abbrüche.

4.5.2 Die Auswertung eines Ausdrucks

Viele Basic-Befehle benötigen zur Ausführung Parameter, die dem Befehlstoken folgen. Diese Parameter können Adressen, Integerwerte, REAL-Zahlen oder Strings sein. Zur Auswertung eines beliebigen Ausdrucks gibt es im Basic-Interpreter eine Routine, die einen beliebig verschachtelten Ausdruck berechnet. Der Ausdruck kann aus Variablen, Konstanten, Funktionen,Operatoren und Klammern bestehen. Diese Routine liegt im CPC 464 bei $CEFB, im CPC 664 bei $CF65 und im CPC 6128 bei $CF62.

Die Routine benutzt zur Speicherung von Zwischenergebnissen und zur Übergabe des berechneten Ausdrucks einen Speicherbereich ab $B0C1 ($B09F im CPC 664/6128), den wir mit FAC (Fließkomma-Akkumulator) abkürzen wollen. Der FAC kann jedoch auch Integerwerte und Strings aufnehmen, im letzteren Fall wird ein Zeiger auf den Stringdescriptor gespeichert. Das erste Byte enthält den Typ des FAC: eine 2 für Integer, eine 3 für String und eine 5 für REAL. Dieses Typ-Byte ist gleichzeitig

die Größe des Integer- oder REAL-Wertes bzw. des Stringdescriptors. Beim Übertragen des FAC in eine Variable gibt das Typflag also an, wieviel Bytes für den Eintrag des Variablenwerts benötigt werden.

Ein Ausdruck wird auf oberster Ebene Operand für Operand geholt. Zwei Operanden werden dann miteinander entsprechend des zwischen ihnen stehenden Operators verknüpft. Operatoren sind hier nicht nur "+", "-", "*", "/", "^", "\" und "MOD", sondern auch "AND", "OR", "XOR" und die Vergleichsoperatoren ">", "=", ">=", "<", "<>" und "<=". Eine Sonderstellung nehmen die Operatoren "NOT" und "-" (als Vorzeichenwechsel, nicht als Subtraktion) ein, die nur einen Operanden benötigen.

Die Routine "Ausdruck auswerten" ruft zunächst eine andere Routine auf, die erst einmal den ersten Operanden berechnet. Die Routine zur Berechnung eines Einzeloperanden prüft das folgende Token; wenn es eine Variable oder Konstante ist, so wird der entsprechende Wert in den FAC geholt. Bei einer Funktion müssen vor ihrer Ausführung erst ein oder mehrere Argumente berechnet werden. Dies geschieht mit Hilfe der Routine "Ausdruck auswerten". Da die Routine "Einzeloperanden berechnen" aber gerade von dieser Routine aufgerufen wurde, handelt es sich hier um eine indirekte Rekursion (siehe auch Kapitel 2.3.1). Nicht nur bei Funktionsargumenten, auch zur Bestimmung von den aktuellen Indizes einer Feldvariablen oder zur Berechnung eines Ausdrucks in Klammern wird "Ausdruck auswerten" indirekt rekursiv aufgerufen. Der erste Einzeloperand kann also aufgrund des rekursiven Aufrufs schon mehrere Operanden und Operatoren enthalten.

Nachdem der erste Operand geholt worden ist, wird das Operator-Token untersucht und dann der zweite Operand geholt. Dieser kann jedoch nicht mit der Routine "Einzeloperanden berechnen" direkt ausgewertet werden, weil er alle stärker bindenden Operatoren enthalten muß. Ein Beispiel zur Verdeutlichung: Der erste Einzeloperand des Ausdrucks "A*B^2+4" lautet "A". Es wird also der Wert der Variablen A geholt. Der zweite Operand für die Multiplikation ist jedoch nicht der Einzeloperand "B", sondern der Operand "B^2", da die Potenzierung eine höhere Priorität besitzt als die Multiplikation. Der zweite Operand muß also inklusive aller stärker bindenden Operatoren berchnet werden. Das Beispiel läßt sich auch auf den Ausdruck "(A+B)*(3-C)^2+4" übertragen, ungeachtet der in den Klammerausdrücken auftretenden Operatoren, die ja rekursiv berechnet werden.

Zur Festlegung, was stärker bindende Operatoren bzw. solche mit höherer Priorität sind, wird jedem Operator ein Hierarchiecode zugeordnet. Je größer der Hierarchiecode, desto höher die Priorität.

Hierarchiecode	Operator
keiner	Stringverknüpfung (+) (höchste Priorität)
$16	Potenzierung (^)
$14	Vorzeichenwechsel (-)
$12	Multiplikation, Division (*,/)
$10	Integerdivision (\)
$0E	Integer-Modulo (MOD)
$0C	Addition, Subtraktion (+,-)
$0A	Vergleich (>,=,>=,<,<>,<=)
$08	Einerkomplement (NOT)
$06	Und-Verknüpfung (AND)
$04	Oder-Verknüpfung (OR)
$02	Exklusiv-Oder-Verknüpfung (XOR)

Es gibt eine Routine "Teilausdruck holen", die einen Hierarchiecode übergeben bekommt. Sie holt den ersten Einzeloperanden, wertet den Operator aus und holt weitere Operanden inklusive stärker bindender Operatoren. Sie bricht dann ab, wenn sie auf einen Operator stößt, dessen Hierarchiecode kleiner oder gleich dem übergebenen Hierarchiecode ist, der also schwächer bindet. Sie holt also einen Teilausdruck inklusive aller stärker bindenden Operatoren. Dies ist genau die Aufgabe, die zur Berechnung des zweiten bzw. weiterer Operanden nötig ist. Die Routine ruft sich daher selbst rekursiv auf.

Ein Problem taucht noch auf: Der erste Operand wird zunächst in den FAC geholt. Er muß zwischengespeichert werden, damit der zweite Operand in den FAC geholt und die Operation ausgeführt werden kann. Da der zweite Operand rekursiv geholt wird, muß der erste Operand auf einem Stack zwischengespeichert werden. Deshalb (und für andere Zwecke) existiert im CPC ein vom Basic verwalteter Software-Stack. Dieser Basic-Stack liegt im Bereich $AE8B-$B08A (beim CPC 664/6128 von $AE6F bis $B06E).

Bei Strings läuft diese Zwischenspeicherung anders: Für die Stringdescriptoren existiert ein eigener Stack von $B09C bis $B0B9 beim CPC 464 bzw. von $B07E bis $B09B beim CPC 664/6128. Der Zeiger auf den aktuellen Descriptor wird aus dem FAC geladen und auf den Hardware-Stack gerettet.

4.5.3 Auswertung von mit DEF FN definierten Funktionen

Das Schneider-Basic bietet dem Benutzer die Möglichkeit, selbst Funktionen zu definieren. Einer so definierten Funktion können kein, ein oder mehrere Argumente übergeben werden, die die Typen Integer, Real oder String haben. Das Funktionsresultat kann ebenfalls von einem beliebigen Typ sein.

Ein Zeiger auf eine Funktionsdefinition wird vom Basic-Interpreter beim DEF FN-Befehl als Intergervariable abgelegt. Zur Unterscheidung von einer normalen Variablen wird das 6. Bit im Typ-Byte gesetzt. Gespeichert wird der Zeiger auf die Definition der formalen Parameter. Falls keine Parameter übergeben werden, wird der Zeiger auf das Gleichheitszeichen vor der Ergebnisdefinition gespeichert.

Bei der Auswertung einer definierten Funktion werden zunächst die aktuellen Parameter geholt und deren Typen und Anzahl mit Typen und Zahl der formalen Parameter verglichen. Die aktuell übergebenen Parameter werden mit Namen, Typ und Wert auf dem Basic-Stack in einer verketteten Liste gespeichert (Abbildung 4.2).

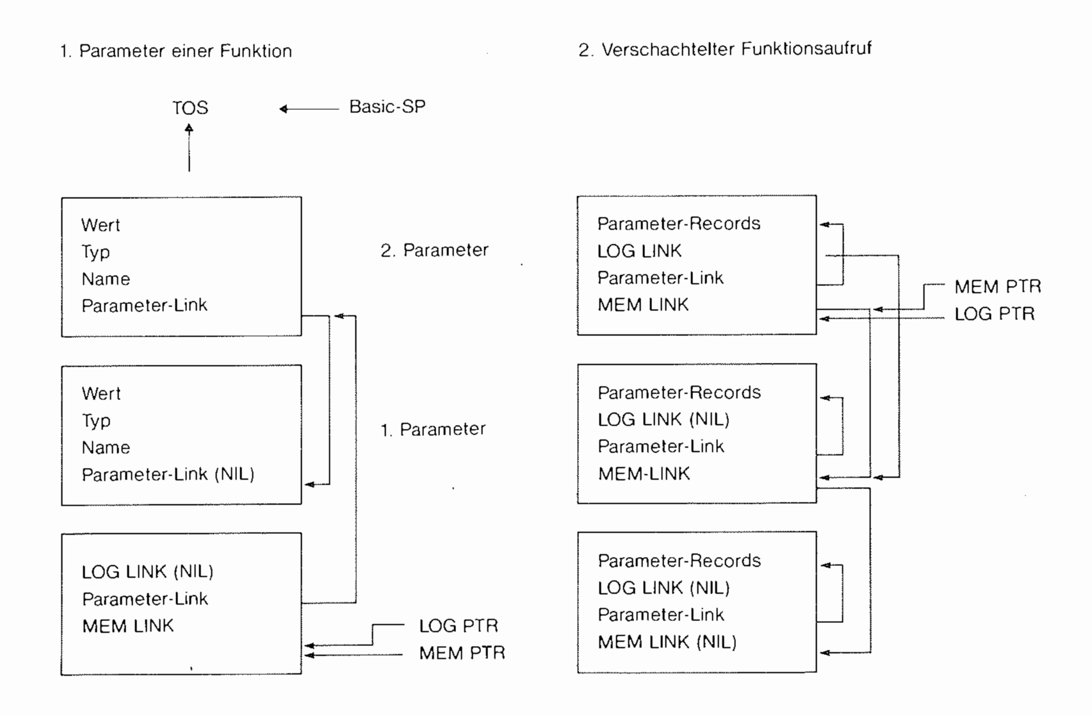


Abbildung 4.2: *Auswertung definierter Funktionen*

Beim Berechnen der aktuellen Parameter kann es erforderlich sein, wiederum eine definierte Funktion auszuwerten. In diesem Fall muß natürlich eine neue verkettete Liste der Parameter auf dem Basic-Stack

angelegt werden. Um die Position des alten Parameter-Records auf dem Stack nicht zu verlieren, werden die Parameter-Records durch eine verkettete Liste verbunden (MEM LINK in Abbildung 4.2). Hier ist also ein Beispiel für einen Stack, der durch eine verkettete Liste realisiert ist (siehe dazu auch Kapitel 2.2).

Wenn alle Parameter für die definierte Funktion geholt worden sind, wird das Funktionsergebnis mit Hilfe der Routine "Ausdruck auswerten" berechnet. Die Namen der formalen Parameter müssen bei dem zu berechnenden Ausdruck durch die aktuell übergebenen Parameter ersetzt werden. Dazu wird das Parameter-Record der definierten Funktion in eine weitere verkettete Liste eingehängt. In Abbildung 4.2 geschieht dies über LOG LINK. Während die oberen beiden Parameter-Records auch in der LOG LINK-Liste eingehängt sind, werden bei der Funktion, die zum unteren Record gehört, gerade die Parameter berechnet. Diese Parameter-Berechnung enthält dann einen Aufruf der mittleren Funktion. Deren Ergebnisberechnung ruft wiederum die obere Funktion auf.

Das Einhängen in eine Liste (bzw. einen Listen-Stack) ist hier ebenfalls erforderlich, weil bei der Ergebnisberechnung Verschachtelungen auftreten können. Der oberste Parameter-Record in der Liste, also der Parameter-Record der in der innersten Verschachtelung gerade bearbeiteten Funktion, wird bei der Variablenauswertung benutzt. Immer, wenn der Wert einer Variablen geholt werden soll, wird zuerst in diesem Parameter-Record nach dem Variablennamen gesucht und gegebenenfalls der aktuelle Parameterwert als Variablenwert zurückgegeben. So werden für die Dauer der Funktions-Ergebnisberechnung die entsprechenden Variablen durch die aktuellen Parameter substituiert. Nach der Ergebnisberechnung wird der Parameter-Record wieder aus der LOG LINK-Liste ausgehängt. Die Variablen liefern dann also wieder ihren ursprünglichen Wert.

4.6 Datenformate auf dem Basic-Stack

Der im vorigen Kapitel zur Sprache gekommene Basic-Stack wird nicht nur zur Ausdruckauswertung benutzt. Auch Schleifen und Unterprogramme werden über ihn realisiert. Es ist die Frage, weshalb ein eigener Software-Stack benutzt wird, dessen Verwaltung ja langsamer ist, als die des Hardware-Stacks. Eine Antwort ist die, daß zur Ablage von Werten auf dem Stack Unterprogramme angesprungen werden können, deren Rückkehradressen bei Ablage eines Wertes auf dem Hardware-Stack gesondert berücksichtigt werden müßten.

4.6.1 Ein FOR-NEXT-Schleifen-Eintrag

Bytes

2	Adresse der FOR-Schleifenvariablen
2/5	Schleifenendwert im REAL bzw. Integer-Format
2/5	Schleifen-STEP-Wert im REAL bzw. Integer-Format
1	Vorzeichen des STEP-Werts
2	Zeiger nach FOR-Statement
2	Zeilenadresse des FOR-Statements
2	Zeiger nach NEXT-Statement
2	Zeiger nach NEXT-Token
1	$10 bei Integerschleifen, $16 bei REAL-Schleifen, als Kennung und als Größe des Stackeintrags

4.6.2 Ein WHILE-WEND-Schleifen-Eintrag

Bytes

2	Zeilenadresse des WHILE-Statements 2
2	Zeiger nach WEND-Token 1
2	Zeiger nach WHILE-Token
1	$07 als Kennung und als Größe des Stackeintrags

4.6.3 Ein GOSUB-Unterprogramm-Eintrag

Bytes

1	0 für einfacher GOSUB-Befehl
2	Zeiger nach GOSUB-Statement als Rückkehradresse
2	Zeilenadresse des GOSUB-Statements
1	$06 als Kennung und als Größe des Stackeintrags

4.6.4 Ein Eintrag bei Unterbrechungen

Bytes

Bytes	
1	Anlaß des Unterprogrammaufrufes 1 = EVERY-, AFTER- oder ON SQ-Unterbrechung 2 = ON BREAK-Unterbrechung
2	Adresse des Event-Block-Parameterfeldes
2	Adresse der Zeile, in der die Unterbrechung auftrat
1	$06 als Kennung und als Größe des Stackeintrags

4.7 Benutzung von Synchronous Events im Basic

Die Bearbeitung von Synchronous Events wurde im Kapitel 3.1.4 bereits erklärt. An dieser Stelle soll ihre spezielle Benutzung im Basic beschrieben werden.

Synchrone Events werden im Basic an mehreren Stellen benutzt: für die Befehle AFTER, EVERY, ON SQ GOSUB und zur Bearbeitung von Unterbrechungen durch ESC. Dies geschieht mit Hilfe von acht Event-Blocks. Vier davon, die "Uhren" 0 bis 3, werden für EVERY und AFTER benutzt; sie sind in der Ticker Chain eingehängt. Den Tick- und Reload-Count kann man durch die an EVERY und AFTER übergebenen Werte setzen (siehe Abschnitt 3.1.4.3). Drei weitere Event-Blocks entsprechen den drei Tonkanälen. Sie sind in keiner Chain, sondern werden vom Sound Manager in die Synchronous Pending Queue eingehängt. Schließlich wird noch der Break-Event-Block des Keyboard Manager benutzt (siehe Kapitel 3.7.3). Der wird immer dann in die SPQ eingehängt, wenn die ESC-Taste gedrückt wurde und der Break-Event mit KM ARM BREAK erlaubt wurde.

Hier die Prioritäten der einzelnen Events:

Break-Event:	$40 (Express-Event)
AFTER/EVERY-Event 3:	$10
AFTER/EVERY-Event 2:	$08
Sound-Events:	$08
AFTER/EVERY-Event 1:	$04
AFTER/EVERY-Event 0:	$02
(Basic-Hauptprogramm:	$00)

Jeder Event-Block im Basic besitzt ein Parameter-Feld, das nach dem eigentlichen Event-Block folgt. Für den Break-Event-Block, der ja im Keyboard Manager liegt, existiert ein separater Parameter-Block. Ein solcher Block ist wie folgt aufgebaut:

0	Priorität des unterbrochenen Events/Programms
1/2	Rücksprung-Basic-Programmzeiger (Basic-PC)
3/4	Adresse des auszuführenden Unterprogramms

Die Interpreterschleife ruft zwischen zwei Statements die Routine KL POLL SYNCHRONOUS auf. Wenn diese Routine die Information zurückgibt, daß ein Event mit einer höheren Priorität als der laufenden in der SPQ ist, so wird eine Routine im Basic aufgerufen, die die entsprechenden Events der SPQ bearbeitet. Zur Bearbeitung werden in einer Schleife die Routinen KL NEXT SYNC, KL DO SYNC und KL DONE SYNC aufgerufen. Dies veranlaßt die Ausführung der Event-Routinen.

Die Bearbeitung der Events geschieht mit Hilfe bestimmter Bearbeitungs-Flags. Diese werden von der von KL DO SYNC ausgeführten Event-Rou-

tine gesetzt, um der aufrufenden Routine (der NEXT-DO-DONE-Schleife) mitzuteilen, was zu tun ist. Diese Flags liegen im CPC 464 bei $AC30, im CPC 664 und im CPC 6128 bei $AC16. Die einzelnen Bits haben folgende Bedeutung:

Bit 0:	1 = Abbruch, "Break" ausgeben
Bit 1:	1 = Basic-Unterprogramm bei ($AE34) ausführen
Bit 2:	1 = Break-Event wieder erlauben (mit KM ARM BREAK)
Bit 6:	1 = NEXT-DO-SYNC-Schleife beenden
Bit 7:	1 = KL DONE SYNC noch nicht aufrufen

Die AFTER/EVERY- und Sound-Events benutzen alle die gleiche Event-Routine. Wenn gerade ein Programm läuft, also der Direkt-Modus nicht eingeschaltet ist, so wird ein Datensatz auf dem Basic-Stack generiert. Außerdem werden die Flags für Abbruch der Schleife, kein KL DONE SYNC und Unterprogramm aufrufen, gesetzt. KL DONE SYNC wird erst beim RETURN-Befehl des Unterprogramms aufgerufen, um die alte Priorität wieder zu setzen.

Eine spezielle Event-Routine wird beim Break-Event ausgeführt. Diese Routine wartet auf eine weitere Taste von der Tastatur. Wenn diese eine zweite ESC-Taste ist - nach der ersten, die den Break-Event auslöste -, so wird entweder - wenn keine ON BREAK-Routine vorhanden ist - ein Flag für Abbruch gesetzt. Andernfalls wird - wie bei AFTER, EVERY und ON SQ - die Ausführung einer Routine vorbereitet. Falls jedoch keine zweite ESC-Taste gedrückt wurde, so setzt die Break-Event-Routine nur ein Flag zur Ausführung von KM ARM BREAK. Damit ist eine erneute Unterbrechung durch ESC möglich. Ansonsten wird im letzteren Fall die Sync-Schleife normal weitergeführt und das Basic-Programm nicht unterbrochen (wenn keine anderen Events in der SPQ sind).

Eine Besonderheit ist, daß die Routine KM ARM BREAK und somit auch die Basic-Routine, die KM ARM BREAK aufruft, den Break-Event nur einmal zuläßt. Nach einem Break-Event muß also KM ARM BREAK erneut aufgerufen werden, wenn man die Funktion der ESC-Taste erhalten will.

Im CPC 664 und im CPC 6128 existiert ein Flag bei $AC0B, das den Aufruf von KM ARM BREAK verhindert. Es wird vom Basic-Befehl ON BREAK CONT gesetzt. Danach wird die Break-Event-Routine gar nicht erst ausgeführt.

4.8 Erweiterungen und Veränderungen des Basic

4.8.1 Die Benutzer-Vektoren des CPC 464

An neun Stellen im Basic-Interpreter des CPC 464 wird eine Routine im Basic-Systembereich angesprungen. Normalerweise steht an den angesprungenen Stellen ein RET, der Ansprung bewirkt also nichts. Es sind jedoch drei Bytes pro Ansprung freigelassen, Platz genug, um einen JP-Opcode ($C3) und eine Zwei-Byte-Adresse einzufügen. Mit anderen Worten: Diese Benutzer-Vektoren kann man - neben der Möglichkeit der RSX-Erweiterung - verwenden, um das Basic selbst zu verändern und zu erweitern. Im CPC 664 und im CPC 6128 sind diese Ansprünge leider entfernt worden, das Basic kann hier nur noch über RSX erweitert werden (siehe 4.8.2).

Hier ist nun eine Tabelle mit den Benutzer-Vektoren des CPC 464. Sie enthält zur besseren Orientierung die Stelle, an der der jeweilige Vektor aufgerufen wird sowie auch einige übergeordnete Aufrufe.

Bedeutung	Vektor	Aufruf-Linie
Eingabeschleife:	$AC01,	$C064
Fehlerbehandlung:	$AC04,	$CA94
Befehl ausführen:	$AC07*,	$DDC3, $DDB1
Funktionsauswertung:	$AC0A*,	$D0A9, $D09A
Einzeloperanden-Auswertung:	$AC0D*,	$D078, $CFF3, $CFE3
Item tokenisieren:	$AC10,	$DEE1, $DEC9
Item wieder nach ASCII:	$AC13,	$E196, $E18D
Variable/Keyword mit Buchst. tokenisieren:	$AC16,	$DF51, $DF09
Keyword zu einem Token suchen:	$AC19*,	$E30B, $E2ED, $E229

Mit einem Stern (*) sind diejenigen Vektoren gekennzeichnet, bei denen nach dem Vektor-Aufruf ein Fehler wegen eines falschen Zeichens oder eines nicht bekannten Tokens an der aktuellen Stelle im Programm ausgegeben wird. Will man neue Tokens benutzen, so muß man zumindest diese Vektoren ändern und die Ausgabe eines Fehlers verhindern.

Die übrigen Vektoren werden zwar nicht kurz bevor der Ausgabe eines Fehlers angesprungen, sind aber dennoch wichtig für die Bearbeitung neuer Befehle und Funktionen mit neuen Tokens. Die Benutzer-Vektoren, die bei Eingabeschleife und Fehlerbehandlung angesprungen werden, eignen sich nicht direkt zur Befehls- oder Funktionserweiterung. Daß sie aber dennoch nützlich sein können, zeigt folgendes Beispielprogramm:

```
10 MEMORY HIMEM-5
20 H=INT(HIMEM+1)
30 FOR I=H TO H+4:READ A:POKE I,A:NEXT
40 POKE &AC02,H AND 255
50 POKE &AC03,(H AND -256)\256 AND 255
```

```
60 POKE &AC01,&C3
70 DATA &AF,&32,&45,&AE,&C9
80 REM Zeile 70:
90 REM XOR A
100 REM LD (AE45),A
110 REM RET
```

Dieses Programm verändert den Vektor für die Eingabeschleife. Bei jedem Einsprung in die Eingabeschleife wird dann das kleine Maschinenprogramm, das ab Zeile 80 disassembliert ist, ausgeführt. Das bedeutet, daß die Speicherstelle $AE45 ständig auf Null gesetzt wird. Da $AE45 beim CPC 464 angibt, ob ein Progamm geschützt ist oder nicht, lassen sich nach der Ausführung des obigen Programms auch geschützte Programme laden und listen. Das Flag, welches das geschützte Programm kennzeichnet, wird sofort nach dem Laden gelöscht. Dieses kleine Programm kann hilfreich sein, wenn man versehentlich nur eine geschützte Version eines Programms abgespeichert hat. Zum Laden von Software, deren Vervielfältigung nicht erlaubt ist, darf dieses Hilfsprogramm jedoch nicht benutzt werden.

4.8.2 RSX-Erweiterungen

Der CPC bietet die Möglichkeit, mit RSX ("resident system extension") das Schneider-Basic um zusätzliche Befehle zu erweitern. Eine Beschreibung des Aufbaus einer RSX-Erweiterung finden Sie in Abschnitt 3.1.3. Hier geht es in erster Linie um die Schnittstelle zum Basic.

Ein RSX-Befehl wird mit einem senkrechten Strich (Shift-@) und einem RSX-Befehlswort aufgerufen. Es können bis zu 32 Parameter übergeben werden. Beim CPC 464 müssen dies Integerwerte sein, beim CPC 664 und CPC 6128 sind dagegen auch Strings zulässig. Der Befehlsroutine wird in IX ein Zeiger auf eine Tabelle mit den Parametern und in A deren Anzahl übergeben. Die Parameter-Tabelle enthält pro Parameter einen Zwei-Byte-Wert. Entweder ist dies der Integerwert oder der Zeiger auf den Stringdescriptor. Ein Unterscheidungsmerkmal zwischen den beiden Parameter-Typen gibt es nicht. Die Parameter sind in der Tabelle in umgekehrter Reihenfolge enthalten, d.h. der letzte Wert in der Parameterliste erscheint in der Parameter-Tabelle an erster Stelle.

Die Parameterübergabe beim Basic-Befehls CALL verläuft übrigens ebenso. Die RSX-Erweiterungen im RAM sind also eigentlich CALL-Befehle, die mit Namen versehen wurden. Beim Aufruf einer Maschinenroutine ist die RSX-Erweiterung dem CALL-Befehl also fast immer vorzuziehen. Ein signifikanter Name ist doch wesentlich bequemer als eine einfache Adresse.

Hier ein einfaches Beispiel für eine RSX-Erweiterung. Das Wort WAITCHAR (vor WAITCHAR natürlich SHIFT-@ eingeben!) ruft die Betriebssystem-Routine KM WAIT CHAR auf, wartet also auf eine Taste.

```
10 MEMORY &7FFF
20 DATA &05,&80                               ' Zeiger auf Befehlstabelle
30 DATA &C3,&18,&BB                           ' JP BB18, KM WAIT CHAR
40 DATA &57,&41,&49,&54,&43,&48,&41,&D2       ' "WAITCHAR"
50 DATA 0                                     ' Tabellenende
60 DATA -1
70 DATA &21,&00,&81                           ' LD HL,8100
80 DATA &01,&00,&80                           ' LD BC,8000
90 DATA &C3,&D1,&BC                           ' JP BCD1
100 DATA -1
110 RESTORE 20:FOR I=&8000 TO &8100:READ A
120 IF A>=0 THEN POKE I,A:NEXT
130 RESTORE 70:FOR I=&8200 TO &8300:READ A
140 IF A>=0 THEN POKE I,A:NEXT
150 CALL &8200
```

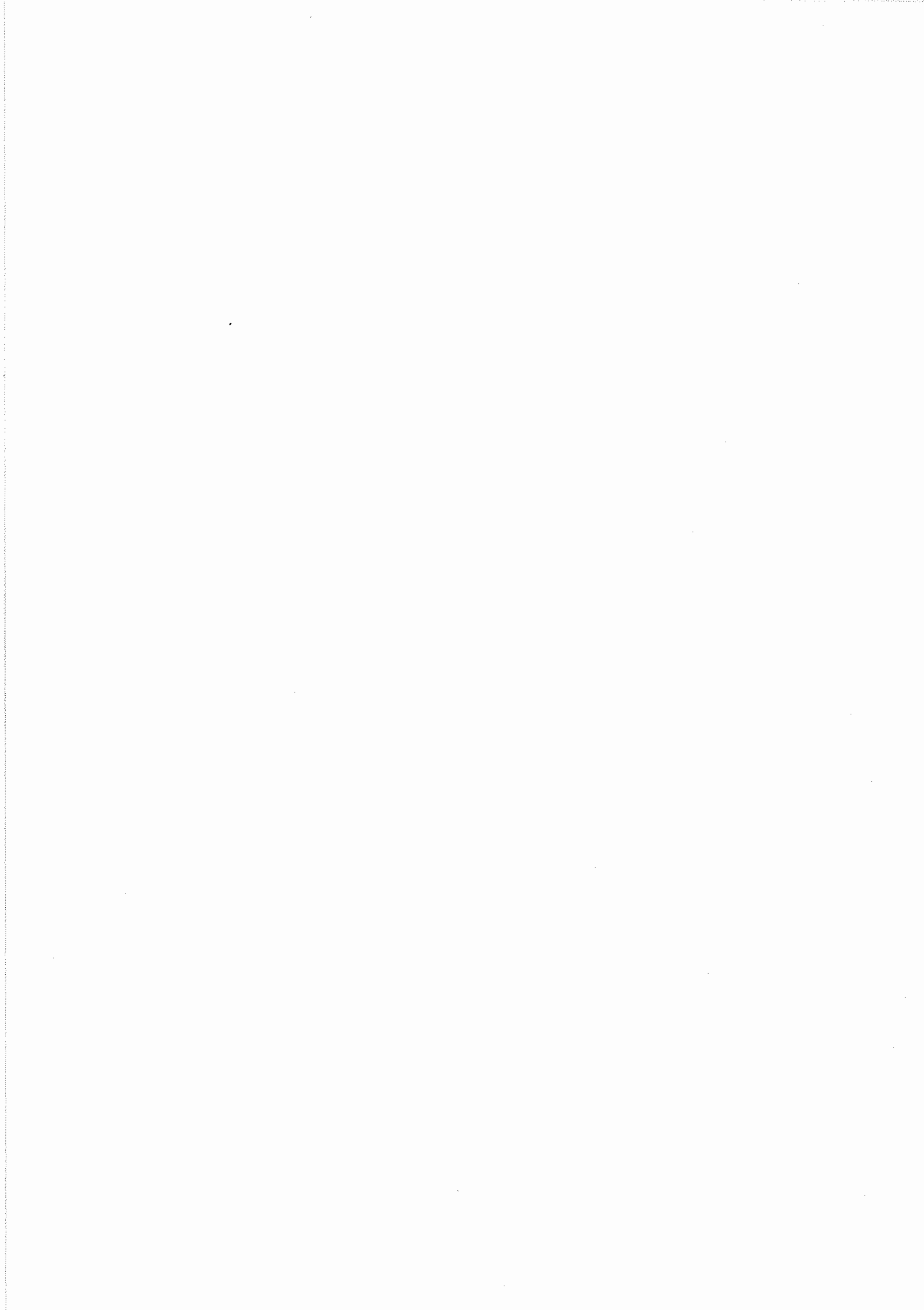
4.9 Ergänzungen zum Handbuch

Im Handbuch des CPC 464 sind einige Möglichkeiten des Schneider-Basics nicht erklärt. Wir wollen diese erläutern.

MID$ kann nicht nur als Funktion, sondern auch als Zuweisung verwendet werden, wenn ein Teil eines Strings durch einen anderen String ersetzt werden soll. Die Anweisungen A$="C 64" und MID$(A$,2,4)="PC 4" ergeben als neuen Wert von A$ den String "CPC 464". Es bietet sich dem Programmierer hier eine komfortable Möglichkeit, die Zerlegung eines Strings und die anschließende Verkettung durch einen Schritt zu ersetzen. Im Handbuch des CPC 664/6128 ist diese Zuweisung übrigens beschrieben.

Die Funktion DEC$ ist ebenfalls nur im Handbuch des CPC 464 nicht beschrieben. Dies hat wohl den Grund, daß nach dem DEC$-Schlüsselwort zwei offene Klammern ("((") folgen müssen. Im CPC 664 und im CPC 6128 ist dieser Fehler behoben worden. Der Zweck der DEC$-Funktion, der ein numerisches Argument und ein String-Argument übergeben werden muß: Sie formatiert eine Zahl entsprechend eines Formatstrings (genau wie bei PRINT USING). Der Ergebnis-String wird jedoch nicht sofort ausgedruckt, sondern zurückübergeben, so daß man ihn vor der Ausgabe noch weiter bearbeiten kann. Ein Beispiel: DEC$(-265.94,"###.#-") ergibt den String "265.9-".

Bei der formatierten Ausgabe mit PRINT USING kann man zusätzliche Zeichen in den Formatstring einfügen, die mit ausgedruckt werden. Dies kann z.B. nützlich sein, wenn man statt der amerikanischen oder englischen die deutsche Währung "DM" an einer festen Stelle in der formatierten Ausgabe einfügen möchte. Dies geschieht etwa auf folgende Weise: PRINT USING "DM ####.##";Betrag. Wenn in diesen direkt auszugebenden Zeichen Formatierzeichen wie "#", "^", "+" usw. enthalten sein sollen, so muß vor jedem dieser Formatierzeichen ein Underscore-Character (SHIFT-0) eingefügt werden, damit die Formatierzeichen als direkt auszugebende Zeichen erkannt werden.



5 Tabellen zu den Listings

5.1 Das Betriebssystem-RAM

5.1.1 Das RAM des KERNEL:

CPC 464	CPC 664	CPC 6128	
B100-B101	B82D-B82E	B82D-B82E	Start der Asynchron Pending Queue
B102-B103	B82F-B830	B82F-B830	Adresse der letzten APQ-Eintrags
B104	B831	B831	Interrupt-Flags, für die Bearbeitung von TC undAPQ b7: 1 = APQ wird bearbeitet b6: 0 = APQ ist leer b0: 1 = Ticker aktiv, TC bearbeiten
B105-B106	B832-B833	B832-B833	Zwischenspeicher für SP im Interrupt
B107-B180	B834-B8AD	B834-B8AD	<frei>
B181-B186	B8AE-B8B3	B8AE-B8B3	Zwischenspeicher für BC, DE, HL
B187-B18A	B8B4-B8B7	B8B4-B8B7	Timer, 4 Bytes
B18B	B8B8	B8B8	Sperrbyte, verhindert Übertrag auf B18C
B18C-B18D	B8B9-B8BA	B8B9-B8BA	Start der FRAME FLY CHAIN
B18E-B18F	B8BB-B8BC	B8BB-B8BC	Start der FAST TICKER CHAIN
B190-B191	B8BD-B8BE	B8BD-B8BE	Start der TICKER CHAIN
B192	B8BF	B8BF	Frequenzteiler für Ticker (300Hz/6=50Hz)
B193-B194	B8C0-B8C1	B8C0-B8C1	Start der Synchron Pending Queue
B195	B8C2	B8C2	lfd. Sperr-Priorität
B196-B1A5	B8C3-B8D2	B8C3-B8D2	Kopie des RSX-Strings (maximal $100 Bytes)
B1A6-B1A7	B8D3-B8D4	B8D3-B8D4	Adresse des Anfangs der VL der Background-ROMs
		B8D5	laufende RAM-Konfiguration
B1A8	B8D5	B8D6	laufende ROM-Konfiguration/Nr.
B1A9-B1AA	B8D6-B8D7	B8D7-B8D8	Einsprung in lfd. ROM
B1AB	B8D8	B8D9	ROM-Konfiguration für Einsprung
B1AC-B1B9			Tabelle der alten Hi-RAM-Zeiger für die Hintergrund-ROMs 1 .. 7
	B8D9-B8F8	B8DA-B8F9	Tabelle der alten Hi-RAM-Zeiger für die Hintergrund-ROMs 0 .. 15
B1BA-B1C7	B8F9-B8FF	B8FA-B8FF	<frei>

5.1.2 Das RAM des MACHINE PACK:

CPC 464	CPC 664/6128	
	B804-B82C	Printer translation table

5.1.3 Das RAM des SCREEN PACK:

CPC 464	CPC 664/6128	
B1C8	B7C3	Nummer des Mode (0..2)

B1C9-B1CA	B7C4-B7C5	SCR OFFSET
B1CB	B7C6	SCR BASE hi
B1CC-B1CE	B7C7-B7C9	Indirection zu Force, AND, OR, XOR
		für Punktsetzen mit SCR WRITE
B1CF-B1D6		Bitmasken für Pixelauswahl (je nach Mode)
B1D7	B7D2	Blink-Zähler für 2. Farbsatz
B1D8	B7D3	Blink-Zähler für 1. Farbsatz
B1D9	B7D4	BORDER-Farbwert für 2. Farbsatz
B1DA-B1E9	B7D5-B7E4	INK-Farbwerte für 2. Farbsatz
B1EA	B7E5	BORDER-Farbwert für 1. Farbsatz
B1EB-B1FA	B7E6-B7F5	INK-Farbwerte für 1. Farbsatz
B1FB	B7F6	Flag für aktuell ausgewählten Farbsatz
		0 = 1. Farbsatz
		$FF = 2. Farbsatz
B1FC	B7F7	$FF = neue Farbwerte in Tabellen
		0 = Farbwerte an Gate Array übergeben
B1FD	B7F8	aktueller Blink-Zähler für Farb-Event
B1FE-B206	B7F9-B801	Farbwechsel-Event-Block
	B802	gerettete Graphik-Pen-Maske
	B803	gerettete Graphik-Linienmaske
B207-B208		X-Offset zu nächstem Byte bei Linie
B209-B20B		<frei>

5.1.4 Das RAM des TEXT SCREEN PACK:

CPC 464	CPC 664/6128	
B20C	B6B5	Nummer des ausgewählten Windows
B20D-B21B	B6B6-B6C3	Parameter-Block für Window 0
B21C-B22A	B6C4-B6D1	Parameter-Block für Window 1
B22B-B239	B6D2-B6DF	Parameter-Block für Window 2
B23A-B248	B6E0-B6ED	Parameter-Block für Window 3
B249-B257	B6EE-B6FB	Parameter-Block für Window 4
B258-B266	B6FC-B709	Parameter-Block für Window 5
B267-B275	B70A-B717	Parameter-Block für Window 6
B276-B284	B718-B725	Parameter-Block für Window 7
B285-B293	B726-B733	Parameter-Block des aktuellen Windows
B285	B726	Cursorzeile (absolut)
B286	B727	Cursorspalte (absolut)
B287	B728	0 = Hardware-Scrolling
B288	B729	obere Windowgrenze
B289	B72A	linke Windowgrenze
B28A	B72B	untere Windowgrenze
B28B	B72C	rechte Windowgrenze
B28C	B72D	Scrolling-Zähler
B28D		Cursor-Flag
		b0: 0 = enabled, 1 = disabled
		b1: 0 = ON, 1 = OFF
B28E		VDU-Flag
		0 = VDU disabled, keine Zeichenausgabe
		sonst VDU enabled
	B72E	Cursor-/VDU-Flag
		b0: 0 = Cursor enabled, 1 = disabled
		b1: 0 = Cursor ON, 1 = Cursor OFF
		B7: 0 = VDU enabled, 1 = VDU disabled
B28F	B72F	PEN-Farbmaske
B290	B730	PAPER-Farbmaske

B291-B292	B731-B732	Indirection für Hintergrundmodus
B293	B733	0 = Ausgabe auf Textposition (TAGOFF)
		sonst Ausgabe auf Graphikposition (TAG)
B294	B734	Nummer des 1. Zeichens der User-Matrizen
B295	B735	0 = keine User-Matrizen
		$FF = User-Matrizen
B296-B297	B736-B737	Adresse der User-Matrizen
B298-B2B7	B738-B757	Raum für ungepackte Zeichenmatrix
B2B8	B758	Zahl der Zeichen im Control-Buffer
B2B9	B759	aktuelles Steuerzeichen
B2BA-B2C2	B75A-B762	Control-Buffer für Zeichen, die auf Steuerzeichen folgen
B2C3-B322	B763-B7C2	Steuerzeichentabelle
		(jeweils Zahl der nachfolgenden Zeichen und Ausführadresse)
B323-B327		<frei>

5.1.5 Das RAM des GRAPHIC SCREEN PACK:

CPC 464	CPC 664/6128	
B328-B329	B693-B694	Origin, X-Wert
B32A-B32B	B695-B696	Origin, Y-Wert
B32C-B32D	B697-B698	Cursorposition, X-Wert
B32E-B32F	B699-B69A	Cursorposition, Y-Wert
B330-B331	B69B-B69C	linke Window-Grenze
B332-B333	B69D-B69E	rechte Window-Grenze
B334-B335	B69F-B6A0	obere Window-Grenze
B336-B337	B6A1-B6A2	untere Window-Grenze
B338	B6A3	PEN-Farbmake
B339	B6A4	PAPER-Farbmaske
B33A-B341		Zwischenspeicher für Zeichenmatrix
B33A-B33B		kleinere Schrittweite für Linie
B33C-B33D		größere Schrittweite für Linie
B33E-B33F		Differenz-Divisionsrest für Linie
B340-B341		kleinere Breite für Linie
B342-B343	B6A5-B6A6	X-Startwert bzw. laufender X-Wert
B344-B345	B6A7-B6A8	Y-Startwert bzw. laufender Y-Wert
B346		$FF = X-Differenz bei Linie ist größer
B347-B34B		<frei>
	B6A9-B6AA	kleinere minus größere Differenz
	B6AB-B6AC	kleinere Differenz
	B6AD	Vorzeichen der X-Differenz
	B6AE	Vorzeichen der Y-Differenz
	B6AF	Flag für größere Differenz
	B6B0-B6B1	größere Breite (Differenz+1)
	B6A5-B6A6	GRA FILL-Bufferadresse
	B6A7-B6A8	restliche Bufferlänge
	B6A9	$FF = Buffer-Platz reicht aus
	B6AA	Farbmaske der Füll-Farbe
	B6AB	$FF = rechte Seite, $00 = linke Seite
	B6AC-B6AD	alte obere Y-Grenze
	B6AE-B6AF	neue untere Y-Grenze
	B6B0-B6B1	neue obere Y-Grenze
	B6B2	Flag für 1. Pixel einer Linie
	B6B3	Linienmaske (MASK)
	B6B4	Hintergrund-Modus
		$FF = transparent

5.1.6 Das RAM des KEYBOARD MANAGER:

CPC 464	CPC 664/6128	
B34C-B39B	B496-B4E5	KEY TRANSLATION TABLE
B39C-B3EB	B4E6-B535	KEY SHIFT TABLE
B3EC-B43B	B536-B585	KEY CONTROL TABLE
B43C-B445	B586-B58F	Key Repeat Table
B446-B4DD	B590-B627	Expansion String Buffer
B4DE	B628	lfd. Exp String Count
B4DF	B629	lfd. Exp String Code
B4E0	B62A	Put Back Buffer
B4E1-B4E2	B62B-B62C	Start des Exp String Buffers
B4E3-B4E4	B62D-B62E	Ende des Exp String Buffers
B4E5-B4E6	B62F-B630	Anfang des freien Exp String Buffers
B4E7	B631	Shift Lock Flag (b7=1 f. Shift Lock)
B4E8	B632	Caps Lock Flag (b7=1 f. Caps Lock)
B4E9	B633	Startwert f. Wiederholungsverzögerung
B4EA	B634	Startwert f. erste Verzögerung
B4EB-B4F4	B635-B63E	Key State Map, Tabelle der neuen Tastenrückmeldungen
B4F5-B4FE	B63F-B648	Tabelle der direkten Zeilenrückmeldungen
B4FF-B508	B649-B652	Tabelle der positiven Zeilenrückmeldungen
B509	B653	lfd. Verzögerungszähler
B50A-B50B	B654-B655	Tastenkoordinaten der höchsten Taste
B50C	B656	Break Flag, 0 = Break-Event wird nicht eingehängt
B50D-B513	B657-B65D	Break Event Block
B514-B53B	B65E-B685	Ringbuffer für Tastenkoordinaten
B53C	B686	Anzahl der freien Einträge im Ringbuffer +1
B53D	B687	Schreibzeiger in Ringbuffer
B53E	B688	Anz. der Einträge im Ringbuffer +1
B53F	B689	Lesezeiger in Ringbuffer
B540	B68A	Anzahl der Einträge im Ringbuffer
B541-B542	B68B-B68C	Adresse der KEY TRANSLATION TABLE
B543-B544	B68D-B68E	Adresse der KEY SHIFT TABLE
B545-B546	B68F-B690	Adresse der KEY CONTROL TABLE
B547-B548	B691-B692	Adresse der Key Repeat Table
B549-B54F		<frei>

5.1.7 Das RAM des SOUND MANAGER

CPC 464	CPC 664/6128	
B550		restliche Aktivitäten
B551	B1ED	alte Aktivitäten
B552	B1EE	laufende Aktivitäten
B553	B1EF	Frequenzteiler (300Hz/3=100Hz)
B554	B1F0	Flag, zeigt an, ob ein Kanal bearbeitet werden muß (in diesem Fall <>0)
B555-B55B	B1F1-B1F7	Sound Event Block
B55C-B59A	B1F8-B236	Parameterblock, Kanal A
B59B-B5D9	B237-B275	Parameterblock, Kanal B
B5DA-B618	B276-B2B4	Parameterblock, Kanal C
B619	B2B5	lfd. Kontrollbyte für PSG
B61A-B709	B2B6-B3A5	Lautstärke-Hüllkurven, Nummer 1 bis 15 belegen je 16 Bytes
B70A-B7F9	B3A6-B495	Ton-Hüllkurven, Nummer 1 bis 15 belegen je 16 Bytes
B7FA-B7FF		<frei>

5.1.8 Das RAM des CASSETTE MANAGER:

CPC 464	CPC 664/6128	
B800	B118	Meldungs-Flag
		$FF = Meldungen unterdrücken
B801	B119	$FF = Meldung in zwei Stücke zerteilt
B802	B11A	Eingabefile-Status
		464 664
		0 0 nicht geöffnet
		1 1 gerade eröffnet
		2 5 zeichenweise Datei
		3 2 Direkt-Datei
		4 3 Abbruch durch ESC
		5 4 CAS CATALOG aktiv
B803-B804	B11B-B11C	Adresse des Eingabebuffers
B805-B806	B11D-B11E	laufender Eingabebufferzeiger
B807-B846	B11F-B15E	Buffer für gesuchten Header
B847	B15F	Ausgabefile-Status
		464 664
		0 0 nicht geöffnet
		1 1 gerade eröffnet
		2 5 zeichenweise Datei
		3 2 Direkt-Datei
		4 3 Abbruch durch ESC
B848-B849	B160-B161	Adresse des Ausgabebuffers
B84A-B84B	B162-B163	laufender Ausgabebufferzeiger
B84C-B88B	B164-B1A3	Buffer für auszugebenden Header
B88C-B8CB	B1A4-B1E3	Buffer für gelesenen Header
B8CC	B1E4	Ein-/Ausgabeflag
		b0: 1 = Eingabe aktiv
		b1: 1 = Ausgabe aktiv
B8CD	B1E5	Block-Kennzeichen
		$2C = Header
		$16 = Daten
B8CE	B1E6	Flanken-Flag für Eingabesignal
		$55 = nicht invertiertes Signal
		$AA = invertiertes Signal
B8CF	B1E7	beim Lesen erkannte Zeitzählergrenze für 2 Flanken (abhängig von der Baudrate)
B8D0	B1E8	zu verzögernder Zeitwert (bitverschobene Ausgabe)
B8D1	B1E9	Baudraten-Korrekturzeitwert
B8D2	B1EA	Baudraten-Hauptzeitwert
B8D3-B8D4	B1EB-B1EC	Prüf-Wort für 256 Bytes
B8D5-B8DB		<frei>

5.1.9 Das RAM des EDITOR:

CPC 464	CPC 664/6128	
B8DC	B114	Flag, =$FF, wenn Copy Cursor=Edit Cursor
B8DD	B115	Insert Flag, =0, wenn Insert Modus an
B8DE-B8DF	B116-B117	Koordinaten des Copy Cursors
B8E0-B8E3		<frei>

5.1.10 Das RAM des FLOATING POINT PACK:

CPC 464	CPC 664/6128	
B8E4-B8E7	B100-B103	Mantisse der RND-Zahl, normalisiert
B8E8-B8EC	B104-B108	FAC1, Zwischenspeicher f. FLO-Zahl
B8ED-B8F1	B109-B10D	FAC2, Zwischenspeicher f. FLO-Zahl
B8F2-B8F6	B10E-B112	FAC3, Zwischenspeicher f. FLO-Zahl
B8F7	B113	RAD/DEG-Flag, =0 f. RAD
B8F8-B8FF		<frei>

5.2 Das BASIC-System-RAM

```
CPC 464      CPC 664/6128

AC00         AC00         Flag für Spaces bei Token-Wandlung
                              0 = alle Spaces werden übernommen
                              sonst: überflüssige Spaces werden unterdrückt
AC01-AC1B                 User-Vektoren:
AC01                      Eingabeschleife
AC04                      Fehlerbehandlung
AC07                      Befehl ausführen
AC0A                      Funktionsauswertung
AC0D                      Operanden-Auswertung
AC10                      Item tokenisieren
AC13                      Item wieder nach ASCII
AC16                      Variable/Keyword mit Buchstaben tokenisieren
AC19                      Keyword zu einem Token suchen
AC1C         AC01         Flag für AUTO aktiv
AC1D-AC1E    AC02-AC03    AUTO-Zeilennummer
AC1F-AC20    AC04-AC05    AUTO-Schrittweite
AC21         AC06         aktuelle Streamnummer
AC22         AC07         aktueller Eingabekanal
AC23         AC08         aktuelle Druckerposition
AC24         AC09         Ausgabebreite (WIDTH) für Drucker
AC25         AC0A         aktuelle Position für Kassette
             AC0B         0 = ON BREAK CONT aktiv
                          $FF = Unterbrechung durch ESC erlaubt
AC26         AC0C         Flag für NEXT-Routine
                              0: Test auf Schleifenende
                              $FF: Schleifendurchlauf
AC27-AC2B    AC0D-AC11    Zwischenspeicher für FOR-Startwert
AC2C-AC2D    AC12-AC13    Zeiger nach zugehörigem NEXT-Token
AC2E-AC2F    AC14-AC15    Zeilenadresse des zugehörigen WENDs
AC30         AC16         Flag für Bearbeitung der synchronous Events:
                              b0: 1 = Abbruch, Ausgabe von "Break"
                              b1: 1 = Basic-Unterprogramm ausführen
                              b2: 1 = ESC-Abbruch ermöglichen
                              b6: 1 = Schleife beenden
                              b7: 0 = KL DONE SYNC aufrufen
AC31-AC35    AC17-AC1B    Parameter-Block für Break-Event
AC31         AC17         Priorität des unterbrochenen Events
AC32-AC33    AC18-AC19    Rücksprung-Basic-PC
AC34-AC35    AC1A-AC1B    Adresse der ON BREAK-Routine
AC36-AC37    AC1C-AC1D    Zeiger auf Routinenadresse im Break-Event-Block
AC38-AC43    AC1E-AC29    Sound-Queue 0, Event-Block
AC3F-AC43    AC25-AC29    zugehöriger Parameter-Block
AC3F         AC25         Priorität des unterbrochenen Events
AC40-AC41    AC26-AC27    Rücksprung-Basic-PC
AC42-AC43    AC28-AC29    Adresse der SQ-Routine
AC44-AC4F    AC2A-AC35    Sound-Queue 1, Event-Block (Aufteilung
AC50-AC5B    AC36-AC41    Sound-Queue 2, Event-Block analog)
AC5C-AC6D    AC42-AC53    Event-Block 0 für EVERY/AFTER
AC5C-AC61    AC42-AC47    zugehöriger Ticker-Chain-Kopf
AC69-AC6D    AC4F-AC53    zugehöriger Parameter-Block
AC69         AC4F         Priorität des unterbrochenen Events
AC6A-AC6B    AC50-AC51    Rücksprung-Basic-PC
AC6C-AC6D    AC52-AC53    Adresse der AFTER- bzw. EVERY-Routine
AC6E-AC7F    AC54-AC65    Event-Block 1 für EVERY/AFTER (Auftei-
```

AC80-AC91	AC66-AC77	Event-Block 2 für EVERY/AFTER lung
AC92-ACA3	AC78-AC89	Event-Block 3 für EVERY/AFTER analog)
ACA4-ADA5	ACA8-AD8B	Buffer für Eingabe und LIST
ADA6-ADA7	AD8C-AD8D	Fehler-Zeilenadresse für ERRL
ADA8-ADA9	AD8E-AD8F	Zeiger auf Anfang des Statements, in dem der letzte Fehler auftrat
ADAA	AD90	Nummer des Fehlers für ERRN
	AD91	Nummer des Ein-/Ausgabefehlers für DERR
ADAB-ADAC	AD92-AD93	Basic-PC nach Abbruch für CONT
ADAD-ADAE	AD94-AD95	Zeilenadresse nach Abbruch für CONT
ADAF-ADB0	AD96-AD97	Adresse der ON ERROR-Routine
ADB1	AD98	$FF = ON ERROR-Routine gerade aktiv
ADB2-ADBA	AD99-ADA1	Parameter für SOUND-Befehl
ADB2	AD99	Kanal-Status
ADB3	AD9A	Nummer der ENV-Folge
ADB4	AD9B	Nummer der ENT-Folge
ADB5-ADB6	AD9C-AD9D	Tonperiode
ADB7	AD9E	Rauschperiode
ADB8	AD9F	Lautstärke
ADB9-ADBA	ADA0-ADA1	Tondauer
ADBB-ADCA	ADA2-ADB1	Parameter für ENV und ENT
ADCB-ADCF	ADB2-ADB6	Zwischenspeicher für Exponenten bei Potenzierung
ADD0-AE03	ADB7-ADEA	1. Offsets der verketteten Variablenlisten für einfache Variablen (26 Offsets für die Anfangsbuchstaben "A" bis "Z")
AE04-AE05	ADEB-ADEC	1. Offset für VL der DEF FN-Funktionen
AE06-AE07	ADED-ADEE	1. Offset für VL der REAL-Felder
AE08-AE09	ADEF-ADF0	1. Offset für VL der Integer-Felder
AE0A-AE0B	ADF1-ADF2	1. Offset für VL der String-Felder
AE0C-AE25	ADF3-AE0C	Typen für Variablen ohne Kennzeichen (26 Typen für Anfangsbuchstaben "A" bis "Z")
AE26	AE0D	0 = Default-Selbstdimensionierung ohne DIM
AE27-AE28	AE0E-AE0F	Zeiger auf gesuchten Variablennamen
AE29-AE2A	AE10-AE11	Zeiger auf bearbeiteten FN-Listeneintrag
AE2B-AE2C	AE12-AE13	Zeiger auf FN-Liste (VL der Funktionsvariablen)
AE2D	AE14	$3B = kein Linefeed nach Eingabe bei INPUT
AE2E-AE2F	AE15-AE16	DATA-Zeilenadresse
AE30-AE31	AE17-AE18	DATA-Zeiger
AE32-AE33	AE19-AE1A	Basic-Stackpointer bei Statementanfang
AE34-AE35	AE1B-AE1C	Zeiger (Basic-PC) auf aktuelles Statement bzw. Zeiger auf Basic-Unterprogramm für Event-Bearbeitung
AE36-AE37	AE1D-AE1E	aktuelle Zeilenadresse
AE38	AE1F	0 = kein Trace, $FF = Trace
AE39	AE20	Flag für Tokenisierung $01 = Befehl mit Zeilennummer $FF = Variablenname
AE3A	AE21	0 = keine Zeilenadressen im Programm
AE3B-AE3C	AE22-AE23	Start-Löschadresse für DELETE
AE3D-AE3E	AE24-AE25	zu löschende Länge für DELETE
AE3F-AE40	AE26-AE27	Startadresse des Programms beim Laden
AE41	AE28	0 = CHAIN, sonst CHAIN MERGE
AE42	AE29	Filetyp
AE43-AE44	AE2A-AE2B	Länge des Programms beim Laden
AE45	AE2C	$FF = Programm im Speicher ist geschützt
AE46-AE6D	AE2D-AE51	Buffer für Wandlung nach binär, BCD und ASCII
AE6E	AE52	Zahl der Nachkommastellen für Formatierung
AE6F	AE53	Zahl der Vorkommastellen für Formatierung
	AE54	Währungs-Zeichen (Dollar oder Pfund) für Formatierung bei PRINT USING

AE70-AE71		Buffer-Endzeiger für Wandlung nach ASCII
AE72-AE73	AE55-AE56	Adresse der CALL-/RSX-Routine
AE74	AE57	ROM-Konfiguration für CALL-/RSX-Aufruf
AE75-AE76	AE58-AE59	Basic-PC bei CALL-/RSX-Aufruf
AE77-AE78	AE5A-AE5B	Stackpointer bei CALL-/RSX-Aufruf
AE79	AE5C	Tabulatorweite (ZONE)
AE7A	AE5D	Flag für keine Ausgabe bei Ende des PRINT USING-Formatstrings
AE7B-AE7C	AE5E-AE5F	HIMEM-Zeiger
AE7D-AE7E	AE60-AE61	Hi-RAM, Ende des freien RAMs
AE7F-AE80	AE62-AE63	Lo-RAM (Token-Buffer-Zeiger)
AE81-AE82	AE64-AE65	Zeiger auf Programmstart
AE83-AE84	AE66-AE67	Zeiger auf Programmende
AE85-AE86	AE68-AE69	Zeiger auf Variablenstart
AE87-AE88	AE6A-AE6B	Zeiger auf Start der Felder
AE89-AE8A	AE6C-AE6D	Zeiger auf Ende der Felder
	AE6E	$FF = Variablenbereich ist geschützt
AE8B-B08A	AE6F-B06E	Basic-Software-Stack
B08B-B08C	B06F-B070	Basic-Stackpointer
B08D-B08E	B071-B072	Start der Strings
B08F-B090	B073-B074	Ende der Strings
B091	B075	Ein-/Ausgabebuffer-Status
		b0: 1 = Eingabebuffer benutzt
		b1: 1 = Ausgabebuffer benutzt
		b2: 1 = Bereich für Ein- und Ausgabebuffer reserviert
B092-B093		Zeiger vor Ein-/Ausgabebuffer
	B076-B077	Zeiger auf Ein-/Ausgabebuffer
B094-B095		Zwischenspeicher für Hi-RAM bei Reservierung des Ein-/Ausgabebuffer-Bereichs
	B078-B079	Zwischenspeicher für HIMEM+1 bei Reservierung des Ein-/Ausgabebuffer-Bereichs
B096-B097		Zwischenspeicher für Hi-RAM bei Reservierung eines User-Matrizen-Bereichs
B098-B099	B07A-B07B	Offset für Stringbereich-Verschiebung beim Ändern von HIMEM
B09A-B09B	B07C-B07D	Stringdescriptor-Stackpointer
B09C-B0B9	B07E-B09B	Stringdescriptor-Stack
B0BA-B0BC	B09C-B09E	aktueller Stringdescriptor
B0BA	B09C	Stringlänge
B0BB-B0BC	B09D-B09E	Stringadresse
B0BD-B0BE		Zeiger auf Ende des Descriptors der höchsten Stringadresse bei Garbage collection
B0BF-B0C0		höchste Stringadresse bei Garbage collection
B0C1-B0C6	B09F-B0A4	Fließkomma-, Integer- oder Stringakkumulator (FAC)
B0C1	B09F	Typ des FAC
		2 = Integer
		3 = String
		5 = Fließkomma (REAL)
B0C2-B0C3	B0A0-B0A1	Integerzahl
B0C2-B0C3	B0A0-B0A1	Zeiger auf Stringdescriptor
B0C2-B0C6	B0A0-B0A4	REAL-Zahl
B0C7-B0FF	B0A5-B0FF	<frei>

5.3 Die Routinen des Betriebssystems

RAM	464	664	6128	Routine
	0000	0000	0000	RST0: System Reset
	0008	0008	0008	RST1: LO JUMP
	000B	000B	000B	LO PCHL
	000E	000E	000E	JP (BC)
	0010	0010	0010	RST2: LO SIDE CALL
	0013	0013	0013	KL SIDE PCHL
	0016	0016	0016	JP (DE)
	0018	0018	0018	RST3: LO FAR CALL
	001B	001B	001B	KL FAR PCHL
	001E	001E	001E	JP (HL)
	0020	0020	0020	RST4: RAM LAM
	0023	0023	0023	KL FAR CALL
	0028	0028	0028	RST5: FIRM JUMP
	0030	0030	0030	RST6: USER
	0038	0038	0038	RST7: INTERRUPT
	003B	003B	003B	EXT INTERRUPT VECTOR
	0044	0044	0044	Restore Hi Kernel Jumps
BCC8	005C	005C	005C	KL CHOKE OFF
	0077	0077	0077	Reset cont'd 2
BD0D	0099	0099	0099	KL TIME PLEASE
BD10	00A3	00A3	00A3	KL TIME SET
	00B1	00B1	00B1	Scan Events
	00E8	00E8	00E8	async., not Express Events einhängen
	010A	010A	010A	async. PQ und Ticker Chain bearbeiten
	0153	0153	0153	KL KICK EVENT
BCD7	0163	0163	0163	KL NEW FRAME FLY
BCDA	016A	016A	016A	KL ADD FRAME FLY
BCDD	0170	0170	0170	KL DELETE FRAME FLY
BCE0	0176	0176	0176	KL NEW FAST TICKER
BCE3	017D	017D	017D	KL ADD FAST TICKER
BCE6	0183	0183	0183	KL DELETE FAST TICKER
	0189	0189	0189	Ticker Chain bearbeiten
BCE9	01B3	01B3	01B3	KL ADD TICKER
BCEC	01C5	01C5	01C5	KL DELETE TICKER
BCEF	01D2	01D2	01D2	KL INIT EVENT
BCF2	01E2	01E2	01E2	KL EVENT
	020A	0209	0209	Event wiederholt ausführen
BCFE	021A	0219	0219	KL DO SYNC
	021C	021B	021B	Event ausführen
	0223	0222	0222	Near Address Routine ausführen
BCF5	0228	0227	0227	KL SYNC RESET
	022F	022E	022E	Synchronous Event einhängen
BCFB	0256	0255	0255	KL NEXT SYNC
BD01	0277	0276	0276	KL DONE SYNC
BCF8	0285	0284	0284	KL DEL SYNCHRONOUS
BD0A	028E	028D	028D	KL DISARM EVENT
BD04	0295	0294	0294	KL EVENT DISABLE
BD07	029B	029A	029A	KL EVENT ENABLE
BCD1	02A1	02A0	02A0	KL LOG EXT
BCD4	02B2	02B1	02B1	KL FIND COMMAND

```
     02F4 02F1 02F1  RSX-String in ROM suchen
BCCB 0329 0326 0326  KL ROM WALK
BCCE 0332 0330 0330  KL INIT BACK
     0363 0369 0369  Koppeladresse suchen
     0373 0379 0379  Add Event
     0382 0388 0388  Delete Event
BD5B           0397  KL RAM SELECT

     0391 0397 03A6  wird ab B900 ins RAM kopiert
bis  0579 057A 0589

     B900 B900 B900  KL U ROM ENABLE
     B903 B903 B903  KL U ROM DISABLE
     B906 B906 B906  KL L ROM ENABLE
     B909 B909 B909  KL L ROM DISABLE
     B90C B90C B90C  KL ROM RESTORE
     B90F B90F B90F  KL ROM SELECT
     B912 B912 B912  KL CURR SELECTION
     B915 B915 B915  KL PROBE ROM
     B918 B918 B918  KL ROM DESELECT
     B91B B91B B91B  KL LDIR
     B91E B91E B91E  KL LDDR
     B921 B921 B921  KL POLL SYNCHRONOUS
          B92A B92A  KL SCAN NEEDED
     B939 B941 B941  INTERRUPT ENTRY CONT'D
     B970 B970 B970  EXT INTERRUPT ENTRY
     B97C B984 B984  KL LO PCHL CONT'D
     B982 B98A B98A  KL LO JUMP CONT'D
     B9A8 B9B0 B9B0  Sprung nach DE ausführen
     B9B1 B9B9 B9B9  KL FAR PCHL
     B9B9 B9C1 B9C1  KL FAR CALL
     B9BF B9C7 B9C7  KL LO FAR CALL CONT'D
     BA10 BA17 BA17  KL SIDE PCHL CONT'D
     BA16 BA1D BA1D  KL LO SIDE CALL CONT'D
     BA2E BA35 BA35  KL FIRM JUMP CONT'D
     BA4A BA51 BA51  KL L ROM ENABLE
     BA54 BA58 BA58  KL L ROM DISABLE
     BA5E BA5F BA5F  KL U ROM ENABLE
     BA68 BA66 BA68  KL U ROM DISABLE
     BA72 BA70 BA70  KL ROM RESTORE
     BA7E BA79 BA79  KL ROM SELECT
     BA83 BA7E BA7E  KL PROBE ROM
     BA8C BA87 BA87  KL ROM DESELECT
     BA92 BA8D BA8D  ROM-Nummer schreiben
     BAA2 BA9D BA9D  KL CURR SELECTION
     BAA6 BAA1 BAA1  KL LDIR
     BAAC BAA7 BAA7  KL LDDR
     BAB2 BAAD BAAD  ROMs transparent abschalten
     BACB BAC6 BAC6  KL RAM LAM,  HL
     BADC BAD7 BAD7  KL RAM LAM,  IX

     0580 057B 0591  RESET CONT'D
     05B4 05AF 05C5  CRTC-Werte, 50Hz
     05C4 05BF 05D5  CRTC-Werte, 60Hz
BD13 05DC 05D7 05ED  MC BOOT PROGRAM
BD16 060B 0609 061F  MC START PROGRAM
     065C 0657 0677  Einschalt-Meldung ausgeben
     066D 066E 0688  Einschaltmeldung
```

	06E8	06E9	06F9	Ladefehler-Meldung ausgeben
	06EB	06EC	06FC	Meldung ausgeben
	06F4	06F5	0705	Ladefehler-Meldung
	0712	0713	0723	Firmennamen-Adresse holen
	0727	0728	0738	Firmennamen
	0775			JP (HL)
BD1C	0776	0766	0776	MC SET MODE
BD22	0786	0776	0786	MC CLEAR INKS
BD25	0799	077C	078C	MC SET INKS
	07AB	079A	07AA	Farbwert in Gate Array schreiben
BD19	07BA	07A4	07B4	MC WAIT FLYBACK
BD1F	07C6	07B0	07C0	MC SCREEN OFFSET
BD28	07E6	07D0	07E0	MC RESET PRINTER
	07EC	07E1	07F1	Indirection für MC WAIT PRINTER
		07E7	07F7	Printer translation table
BD58		07FC	080C	MC PRINT TRANSLATION
BD2B	07F2	080B	081B	MC PRINT CHAR
	07F8	0825	0835	MC WAIT PRINTER
BD31	0807	0834	0844	MC SEND PRINTER
BD2E	081B	0848	0858	MC BUSY PRINTER
BD34	0826	0853	0863	MC SOUND REGISTER
	0846	0873	0883	Scan Keyboard

BD37	0888	08BB	08BD	JUMP RESTORE
	08AC	08DC	08DE	Default-Adressentabelle
	0A8A	0AB0	0AB4	Indirections kopieren

BBFF	0AA0	0ABB	0ABF	SCR INITIALIZE
BC02	0AB1	0ACC	0AD0	SCR RESET
BC0E	0ACA	0AE5	0AE9	SCR SET MODE
BC11	0AEC	0B08	0B0C	SCR GET MODE
	0AF2	0B0E	0B12	Mode 1 einschalten
BC14	0AF7	0B13	0B17	SCR MODE CLEAR
	0B11	0B2D	0B31	Bitmasken laden, Mode einschalten
BC05	0B3C	0B33	0B37	SCR SET OFFSET
BC08	0B45	0B38	0B3C	SCR SET BASE
BD55		0B41	0B45	SCR SET POSITION
BC0B	0B50	0B52	0B56	SCR GET LOCATION
BC17	0B57	0B59	0B5D	SCR CHAR LIMITS
BC1A	0B64	0B66	0B6A	SCR CHAR POSITION
	0B95	0B97	0B9B	Window-Parameter berechnen
BC1D	0BA9	0BAB	0BAF	SCR DOT POSITION
		0BF2	0BF6	Masken für Pixel holen
BC20	0BF9	0C01	0C05	SCR NEXT BYTE
BC23	0C05	0C0D	0C11	SCR PREV BYTE
BC26	0C13	0C1B	0C1F	SCR NEXT LINE
BC29	0C2D	0C35	0C39	SCR PREV LINE
BC59	0C49	0C51	0C55	SCR ACCESS
	0C68	0C6D	0C71	SCR WRITE
BC5C	0C6B	0C70	0C74	SCR PIXELS, Pixles forcieren
	0C72	0C76	0C7A	XOR-Verknüpfung, Pixels invertieren
	0C77	0C7B	0C7F	AND-Verknüpfung, Pixels löschen
	0C7D	0C81	0C85	OR-Verknüpfung, Pixels setzen
	0C82	0C86	0C8A	SCR READ
BC2C	0C86	0C8A	0C8E	SCR INK ENCODE
BC2F	0CA0	0CA3	0CA7	SCR INK DECODE
	0CAC	0CAE	0CB2	Farbstift-Nummer berechnen
	0CC2	0CC4	0CC8	Farbstiftnummer konvertieren

```
     0CD2 0CD4 0CD8  Farbtabellen initialisieren
BC3E 0CE4 0CE6 0CEA  SCR SET FLASHING
BC41 0CE8 0CEA 0CEE  SCR GET FLASHING
BC32 0CEC 0CEE 0CF2  SCR SET INK
BC38 0CF1 0CF3 0CF7  SCR SET BORDER
     0D0A 0D0C 0D10  Adresse in Farbmatrix berechnen
BC35 0D14 0D16 0D1A  SCR GET INK
BC3B 0D19 0D1B 0D1F  SCR GET BORDER
     0D2F 0D31 0D35  Farbtabellenadressen holen
     0D3C 0D3E 0D42  Farbwechsel-Event-Block einhängen
     0D4F 0D51 0D55  Farbwechsel-Event-Block aushängen
     0D5B 0D5D 0D61  Farbwechsel-Event-Routine
     0D6D 0D6F 0D73  Farbsätze wechseln
     0D81 0D83 0D87  Parameter des lfd. Farbsatzes holen
     0D93 0D95 0D99  Tabelle der Farbwerte entsprechend Nr.
BC44 0DB3 0DB5 0DB9  SCR FILL BOX
BC47 0DB7 0DB9 0DBD  SCR FLOOD BOX
BC4A 0DDF 0DE1 0DE5  SCR CHAR INVERT
BC4D 0DFA 0DFC 0E00  SCR HARDWARE ROLL
     0E24 0E26 0E2A  Textzeile teilweise füllen
     0E37 0E39 0E3D  Offset zu SCR OFFSET addieren
BC50 0E3E 0E40 0E44  SCR SOFTWARE ROLL
     0EA4 0EA6 0EAA  Rasterzeile kopieren
     0EE6 0EE8 0EEC  Test auf Übertrag zu RA-Bits
BC53 0EF3 0EF5 0EF9  SCR UNPACK
BC56 0F49 0F26 0F2A  SCR REPACK
BC5F 0FC4 0F8F 0F93  SCR HORIZONTAL
BC62 102F 0F97 0F9B  SCR VERTICAL
          0FA9 0FAD  Pen- und Linienmaske retten/setzen
          0FBE 0FC2  horizontale Linie ziehen
          1012 1016  vertikale Linie ziehen
          1037 103B  Linien-Parameter berechnen
     104D 104E 1052  Default-Farbwerte

BB4E 1078 1070 1074  TXT INITIALIZE
BB51 1088 1080 1084  TXT RESET
     10A3 109B 109F  alle Windows entsprechend aktuelle Window setzen
     10B7 10AF 10B3  alle Farben decodieren
     10D5 10CD 10D1  alle Farben codieren, Windows auf Default setzen
BBB4 10E8 10E0 10E4  TXT STR SELECT
BBB7 1107 10FF 1103  TXT SWAP STREAMS
     1122 111A 111E  Window-Parameter kopieren
     112A 1122 1126  Adresse der Window-Parameter holen
     113D 1135 1139  Window-Default-Parameter setzen
BB6F 115E 1156 115A  TXT SET COLUMN
BB72 1169 1161 1165  TXT SET ROW
BB75 1174 116C 1170  TXT SET CURSOR
BB78 1180 1178 117C  TXT GET CURSOR
     118A 1182 1186  relative in absolute Position wandeln
     1197 118F 1193  absolute in relative Position wandeln
     11A8 11A0 11A4  Cursor invertieren, Position prüfen
     11AB 11A3 11A7  Cursorposition prüfen, ggf. scrollen
BB87 11CE 11C6 11CA  TXT VALIDATE
     11DA 11D2 11D6  Position in Grenzen forcieren
BB66 120C 1204 1208  TXT WIN ENABLE
     1244 123C 1240  Spaltengrenze in zulässigen Bereich bringen
     124D 1245 1249  Zeilengrenze in zulässigen Bereich bringen
BB69 1256 124E 1252  TXT GET WINDOW
```

	1263	125B	125F	TXT DRAW CURSOR
	1263	125B	125F	TXT UNDRAW CURSOR
BB8A	1268	1261	1265	TXT PLACE CURSOR
BB8D	1268	1261	1265	TXT REMOVE CURSOR
BB81	1279	1272	1276	TXT CUR ON
BB84	1281	127A	127E	TXT CUR OFF
BB7B	1289	1282	1286	TXT CUR ENABLE
BB7E	129A	1293	1297	TXT CUR DISABLE
BB90	12A9	12A2	12A6	TXT SET PEN
BB96	12AE	12A7	12AB	TXT SET PAPER
BB93	12BD	12B6	12BA	TXT GET PEN
BB99	12C3	12BC	12C0	TXT GET PAPER
BB9C	12C9	12C2	12C6	TXT INVERSE
BBA5	12D3	12D0	12D4	TXT GET MATRIX
BBA8	12F1	12FE	1302	TXT SET MATRIX
BBAB	12FD	12FA	12FE	TXT SET M TABLE
BBAE	132A	1327	132B	TXT GET M TABLE
BB5D	1334	1331	1335	TXT WR CHAR
	134A	1347	134B	TXT WRITE CHAR
	1376	1373	1377	Textzeichen-Byte in Bildschirm speichern
BB9F	137A	1377	137B	TXT SET BACK
BBA2	1387	1384	1388	TXT GET BACK
	1391	138E	1392	Byte setzen, Kopie der Matrix
	139F	139C	13A0	Byte setzen, OR-Verknüpfung
BB63	13A7	13A4	13A8	TXT SET GRAPHIC
BB60	13AB	13A8	13AC	TXT RD CHAR
	13C0	13BA	13BE	TXT UNWRITE
	13E3	13DD	13E1	gepackte Matrix suchen
BB5A	1400	13FA	13FE	TXT OUTPUT
	140C	1406	140A	TXT OUT ACTION
BB57	144B	144E	1452	TXT VDU DISABLE
BB54	1451	1455	1459	TXT VDU ENABLE
BD40		145C	1460	TXT ASK STATE
	145B	1460	1464	Steuerzeichentabelle initialisieren
	146B	1470	1474	Steuerzeichentabelle (Default-Tabelle)
BBB1	14CB	14D0	14D4	TXT GET CONTROLS
	14CF	14D4	14D8	SOUND QUEUE-Parameter für CHR$(7)
	14D8	14DD	14E1	CHR$(7), Ton erzeugen
	14E3	14E8	14EC	CHR$(22), Hintergrundmodus setzen
	14E8	14ED	14F1	CHR$(28), Farbstift setzen
	14F1	14F6	14FA	CHR$(29), Rand setzen
	14F8	14FD	1501	CHR$(26), Window definieren
	1504	1509	150D	CHR$(25), Zeichenmatrix definieren
		150F	1513	CHR$(0)
	150A	1515	1519	CHR$(8), Cursor left
	150F	151A	151E	CHR$(9), Cursor right
	1514	151F	1523	CHR$(10), Cursor down/Line feed
	1519	1524	1528	CHR$(11), Cursor up
	152A	1535	1539	CHR$(30), Cursor home
	1530	153B	153F	CHR$(13), Carriage return
	1538	1543	1547	CHR$(31), Cursorposition setzen
BB6C	1540	154B	154F	TXT CLEAR WINDOW, CHR$(12)
	154F	155A	155E	CHR$(16), Zeichen unter dem Cursor löschen
	1556	1561	1565	CHR$(20), Bildschirm ab Cursor löschen
	156D	1574	1578	CHR$(19), Bildschirm bis Cursor löschen
	1584	158B	158F	CHR$(18), Zeile ab Cursor löschen
	158E	1595	1599	CHR$(17), Zeile bis Cursor löschen

BBBA	15B0	15A4	15A8	GRA INITIALIZE
	15D6	15CA	15CE	Graphik-Pen und Paper decodieren
BBBD	15DF	15D3	15D7	GRA RESET
BD43		15E8	15EC	GRA DEFAULT
BBC3	15F1	15F7	15FB	GRA MOVE RELATIVE
BBC0	15F4	15FA	15FE	GRA MOVE ABSOLUTE
BBC6	15FC	1602	1606	GRA ASK CURSOR
BBC9	1604	160A	160E	GRA SET ORIGIN
BBCC	1612	1618	161C	GRA GET ORIGIN
	161A	1620	1624	reale Cursorkoordinaten holen
	161D	1623	1627	Cursor setzen, reale Koordinaten holen
BD4F		1626	162A	GRA FROM USER
	1657	1659	165D	Cursor-relative in absolute Koordinaten umrechnen
	1664			Test, ob Koordinaten für vertikale Linie innerhalb Grenzen sind
	16B0			Test, ob Koordinaten für horizont. Linie innerhalb Grenzen sind
		1666	166A	Test, ob X-Koordinate im Window
		167C	1680	Test, ob Y-Koordinate im Window
	16FC	1690	1694	Test, ob Koordinaten innerhalb Grenzen sind
BBCF	1734	16A1	16A5	GRA WIN WIDTH
	1760	16CD	16D1	X-Grenze in zulässige Grenzen bringen
	1770	16DD	16E1	Koordinaten halbieren
BBD2	1779	16E6	16EA	GRA WIN HEIGHT
	1792	16FF	1703	Y-Grenze in zulässigen Bereich bringen
BBD5	17A6	1713	1717	GRA GET WINDOW WIDTH
BBD8	17BC	1729	172D	GRA GET WINDOW HEIGHT
BBDB	17C5	1732	1736	GRA CLEAR WINDOW
BBDE	17F6	1763	1767	GRA SET PEN
BBE4	17FD	176A	176E	GRA SET PAPER
BBE1	1804	1771	1775	GRA GET PEN
BBE7	180A	1776	177A	GRA GET PAPER
BBED	1810	177C	1780	GRA PLOT RELATIVE
BBEA	1813	177F	1783	GRA PLOT ABSOLUTE
	1816	1782	1786	GRA PLOT
BBF3	1824	1790	1794	GRA TEST RELATIVE
BBF0	1827	1793	1797	GRA TEST ABSOLUTE
	182A	1796	179A	GRA TEST
BBF9	1836	17A2	17A6	GRA LINE RELATIVE
BBF6	1839	17A5	17A9	GRA LINE ABSOLUTE
BD4C		17A8	17AC	GRA SET LINE MASK
BD49		17AC	17B0	GRA SET FIRST
	183C	17B0	17B4	GRA LINE
		1887	188B	reale Cursorkoordinaten als Startkoordinaten setzen
		1894	1898	Teilline für GRA LINE ziehen
		189A	189E	vertikale Teillinie ziehen
		18E7	18EB	horizontale Teillinie ziehen
		1935	1939	Zweierkomplement von HL bilden
BBFC	1945	193C	1940	GRA WR CHAR
	19CF	19C0	19C4	Pixel für Buchstaben setzen
BD46		19D1	19D5	GRA SET BACK
BD52		19D5	19D9	GRA FILL
		1A4C	1A50	angrenzende Linien bearbeiten
		1A99	1A9D	Parameter für Linie in Buffer speichern
		1AC7	1ACB	Parameter für Linie nach anderer Seite in Buffer
		1AE3	1AE7	Linie nach oben ziehen, bis Sperrfarbe
		1AED	1AF1	nach oben gehen, bis Nicht-Sperrfarbe
		1B11	1B15	Linie nach unten ziehen, bis Sperrfarbe
		1B30	1B34	Test, ob Pixel in Sperrfarbe gesetzt ist
		1B3E	1B42	Test, ob Pixel in Sperrfarbe gesetzt ist

		1B4B	1B4F	aktuelle und Grenz-Position berechnen
BB00	19E0	1B5C	1B5C	KM INITIALIZE
BB03	1A1E	1B98	1B98	KM RESET
	1A36			Indirection für KM TEST BREAK
		1BB3	1BB3	Indirections des KM
BB06	1A3C	1BBF	1BBF	KM WAIT CHAR
BB09	1A42	1BC5	1BC5	KM READ CHAR
	1A75	1BF8	1BF8	Put Back Buffer löschen
BB0C	1A77	1BFA	1BFA	KM CHAR RETURN
BD3D		1BFE	1BFE	KM FLUSH
BB15	1A7B	1C04	1C04	KM EXP BUFFER RESET
	1A81	1C0A	1C0A	KM EXP BUFFER CONT'D
	1AB3	1C3C	1C3C	Default Expansion Strings
BB0F	1ABD	1C46	1C46	KM SET EXPAND
	1AE5	1C6A	1C6A	Platz für neuen String schaffen
	1B22	1CA7	1CA7	Anzahl der zu verschiebenden Bytes berechnen
BB12	1B2E	1CB3	1CB3	KM GET EXPAND
	1B3E	1CC3	1CC3	Adresse eines Expansion Strings berechnen
BB18	1B56	1CDB	1CDB	KM WAIT KEY
BB1B	1B5C	1CE1	1CE1	KM READ KEY
	1BA0	1D25	1D25	Keynummer nach ASCII wandeln
BB21	1BB3	1D38	1D38	KM GET STATE
BD3A		1D3C	1D3C	KM SET LOCKS
	1BB7	1D40	1D40	Update Key State Map/MC SCAN KEYS
	1C18	1DA1	1DA1	Tastenkoordinaten in Buffer schreiben
	1C2F	1DB8	1DB8	KM TEST BREAK
	1C48	1DD1	1DD1	Tasten einer Zeile in Buffer schreiben
BB24	1C5C	1DE5	1DE5	KM GET JOYSTICK
BB42	1C69	1DF2	1DF2	KM GET DELAY
BB3F	1C6D	1DF6	1DF6	KM SET DELAY
BB45	1C71	1DFA	1DFA	KM ARM BREAK
BB48	1C82	1E0B	1E0B	KM DISARM BREAK
BB4B	1C90	1E19	1E19	KM BREAK EVENT
BB3C	1CA6	1E2F	1E2F	KM GET REPEAT
BB39	1CAB	1E34	1E34	KM SET REPEAT
BB1E	1CBD	1E45	1E45	KM TEST KEY
	1CCD	1E55	1E55	Adresse und Bitmaske holen
	1CED	1E75	1E75	Ringbuffer initialisieren
	1CFE	1E86	1E86	Eintrag in Buffer schreiben
	1D15	1E9D	1E9D	Eintrag aus Buffer lesen
	1D2C	1EB4	1EB4	Zeiger in Ringbuffer berechnen
BB2A	1D3E	1EC4	1EC4	KM GET TRANSLATE
BB30	1D43	1EC9	1EC9	KM GET SHIFT
BB36	1D48	1ECE	1ECE	KM GET CTRL
	1D4B	1ED1	1ED1	Tastencode holen
BB27	1D52	1ED8	1ED8	KM SET TRANSLATE
BB2D	1D57	1EDD	1EDD	KM SET SHIFT
BB33	1D5C	1EE2	1EE2	KM SET CTRL
	1D5F	1EE5	1EE5	Tastencode setzen
	1D69	1EEF	1EEF	KEY TRANSLATION TABLE
	1DB9	1F3F	1F3F	KEY SHIFT TABLE
	1E09	1F8F	1F8F	KEY CTRL TABLE
BCA7	1E68	1FE9	1FE9	SOUND RESET
	1E9A	201A	201A	Parameter-Blöcke initialisieren
BCB6	1ECB	2050	2050	SOUND HOLD
BCB9	1EE6	206B	206B	SOUND CONTINUE

```
     1F03 208B 208B  Sound Event (asynchronous)
     1F61 20D7 20D7  Scan Sound Queues
BCAA 1F9F 2114 2114  SOUND QUEUE
     203A 219C 219C  Datenblock Adresse berechnen
BCB3 204A 21AC 21AC  SOUND RELEASE
BCAD 206C 21CE 21CE  SOUND CHECK
BCB0 2089 21EB 21EB  SOUND ARM EVENT
          2209 2209  Adresse eines Parameter-Blocks berechnen
     20A8 2213 2213  nächsten Queue-Eintrag holen
     20B7 2219 2219  Kanal aktivieren
     2118 2280 2280  Queue in Haltezustand setzen
          2286 2286  Kanal aus lfd. Aktivitäten löschen
     211F 2290 2290  Rendezvous-Bits auswerten
     2175 22DE 22DE  Tondauer, Rauschen und ENV setzen
     21B2 231B 231B  Default ENV-Folge
     21B6 231F 231F  laufende ENV-Gruppe bearbeiten
     21BF 232B 232B  ENV-Gruppe bearbeiten
     2213 237B 237B  PSG-Hüllkurve setzen
     223A 23A2 23A2  Kanal deaktivieren
     2246 23AE 23AE  nächste ENV-Gruppe setzen
     225F 23C7 23C7  laufende ENV-Kurve initialisieren
     2265 23CD 23CD  ENV-Kurve initialisieren
     2273 23DB 23DB  Lautstärke setzen
     227F            Kanal ausschalten, aus Aktivität herausnehmen
          23E7 23E7  Kanal abschalten
     228B 23E8 23E8  Kanal an/aus, Rauschen an/aus
     22AB 2408 2408  ENT-Ende bearbeiten
     22C2 241F 241F  laufende ENT-Gruppe bearbeiten
     22CB 2428 2428  ENT-Gruppe bearbeiten
     2303 2460 2460  nächste ENT-Gruppe aktivieren
     2313 2470 2470  nächste ENT-Gruppe setzen
     2324 2481 2481  Periodendauer setzen
BCBC 2338 2495 2495  SOUND AMPL ENVELOPE
BCBF 233D 249A 249A  SOUND TONE ENVELOPE
     2340 249D 249D  Hüllkurve kopieren
BCC2 2349 24A6 24A6  SOUND A ADDRESS
BCC5 234E 24AB 24AB  SOUND T ADDRESS
     2351 24AE 24AE  Adresse der Hüllkurve berechnen

BC65 2370 24BC 24BC  CAS INITIALIZE
BC68 237F 24CE 24CE  CAS SET SPEED
BC6B 238E 24E1 24E1  CAS NOISY
BC77 2392 24E5 24E5  CAS IN OPEN
BC8C 23AB 24FE 24FE  CAS OUT OPEN
BC7A 23FC 2550 2550  CAS IN CLOSE
BC7D 2401 2557 2557  CAS IN ABANDON
          256D 256D  Ein-/Ausgabebuffer ggf. löschen
BC8F 2415 257F 257F  CAS OUT CLOSE
BC92 242E 2599 2599  CAS OUT ABANDON
BC80 2435 25A0 25A0  CAS IN CHAR
BC95 245B 25C6 25C6  CAS OUT CHAR
     248B 25F6 25F6  Eingabestatus setzen
     248E 25F9 25F9  Status setzen
BC89 2496 2603 2603  CAS TEST EOF
BC86 249A 2607 2607  CAS RETURN
BC83 24AB 2618 2618  CAS IN DIRECT
     24CF 263C 263C  Buffer kopieren
BC98 24EA 2653 2653  CAS OUT DIRECT
```

```
BC9B 2528 2692 2692  CAS CATALOG
     253F 26AC 26AC  Block von Kassette lesen
     25A8 271A 271A  Lesefehler auswerten
     25B5 2727 2727  falschen Block auswerten
     25C5 2737 2737  Namen und Block vergleichen
     25F3 2760 2760  Namen vergleichen
     260B 2778 2778  Abbruch behandeln
     260D 27AF 27AF  Motor ausschalten
     2614 2786 2786  Block auf Kassette schreiben
     2666 27D8 27D8  Fehler/Abbruch bei Ausgabe auswerten
     2673 27E5 27E5  Meldung ausgeben, Motor ein
     2688 27FA 27FA  ggf. auf Abbruch testen
     2695 2807 2807  Meldung, Namen und "block xxx" ausgeben
     270C 287E 287E  Meldung und CR ausgeben
     2711 2883 2883  Meldung am linken Rand & CR ausgeben
     2713 2885 2885  Meldung und Fehlernummer ausgeben
     271F 2891 2891  Meldung am linken Rand ausgeben
     2726 2898 2898  Meldung ausgeben
     2743 28B5 28B5  Wort und Space ausgeben
     2760 28D2 28D2  Meldung ausgeben, auf Taste warten
     2783 28F3 28F3  Cursor auf 1. Spalte setzen
     278D 28FD 28FD  ggf. Cursor auf nächste Zeile setzen
     27A4 2914 2914  Dezimalzahl ausgeben
     27B6 2926 2926  auf Großschrift forcieren
     27BF 292F 292F  Catalog-Flag holen
     27C5 2935 2935  Kassetten-Meldungen
BCA1 2836 29A6 29A6  CAS READ
BC9E 283F 29AF 29AF  CAS WRITE
BCA4 2851 29C1 29C1  CAS CHECK
     2873 29E3 29E3  Motor ein, Tastatur vorbereiten
     289D 2A0D 2A0D  Block pageweise bearbeiten
     28B8 2A28 2A28  eine Page lesen
     28C7 2A37 2A37  eine Page lesen und vergleichen
     28F7 2A67 2A67  eine Page auf Band schreiben
     2919 2A89 2A89  auf Synchronisation warten
     2923 2A93 2A93  Synchronisation lesen/auswerten
     2964 2AD4 2AD4  Synchronisation schreiben
     2979 2AE9 2AE9  Blockendton schreiben
     2990 2B00 2B00  Bit in Check-Word 'reinmurksen
     29A6 2B16 2B16  Check-Word holen
     29B0 2B20 2B20  Byte einlesen
     29CD 2B3D 2B3D  auf nächste Flanke warten, ESC testen
     29D4 2B44 2B44  auf nächste Flanke warten
     29DD 2B4D 2B4D  auf 2. Flanke im Bit warten
     29F8 2B68 2B68  Byte ausgeben
     2A08 2B78 2B78  Bit auf Band schreiben
     2A37 2BA7 2BA7  nächste Halbwelle ausgeben
BC6E 2A4B 2BBB 2BBB  CAS START MOTOR
BC71 2A4F 2BBF 2BBF  CAS STOP MOTOR
BC74 2A51 2BC1 2BC1  CAS RESTORE MOTOR
     2A72 2BE2 2BE2  Bandhochlaufzeit verzögern, Abbruch testen
     2A89 2BF9 2BF9  Verzögerung für Bandhochlaufzeit
```

464 RAM	464 ROM	664 RAM	664 ROM	6128 RAM	6128 ROM	
BD3A	2A98	BD5B	2C02	BD5E	2C02	EDIT
			2C42		2C42	Zeichen in Buffer übernehmen
	2AC6		2C48		2C48	Editor-Routine anspringen
	2AE0		2C72		2C72	1. Editor-Sprungtabelle
	2B1C		2CAE		2CAE	2. Editor-Sprungtabelle
	2B2B		2CFF		2CFF	BELL
	2B2F		2CBD		2CBD	CRSR UP
	2B33		2CC1		2CC1	CRSR DWN
	2B37		2CC5		2CC5	CRSR RGHT
	2B3B		2CC9		2CC9	CRSR LEFT
	2B42		2CD0		2CD0	ESC
	2B69		2CF3		2CF2	ENTER
	2B75		2D03		2D02	CUR RGHT, Buffer<>0
	2B7E		2D0B		2D0A	CUR DOWN, Buffer<>0
	2B89		2D15		2D14	CTRL-CUR RGHT
	2B92		2D1E		2D1D	CTRL-CUR DOWN
	2B93		2D1F		2D1E	Zeichen in Buffer einfügen
	2BAA		2D35		2D34	CUR LEFT, Buffer<>0
	2BB3		2D3D		2D3C	CUR UP, Buffer<>0
	2BBD		2D46		2D45	CTRL-CUR LEFT
	2BC7		2D50		2D4F	CTRL-CUR UP
	2BC8		2D51		2D50	Cursor in Buffer zurück
	2BEB		2D74		2D73	Windowbreite und Cursorspalte holen
	2BF9		2D82		2D81	CTRL-TAB
	2C01		2D8A		2D8A	Zeichen in Buffer schreiben
	2C17		2D9F		2D9F	Zeichen in Buffer einfügen
	2C3D		2DC3		2DC3	DEL
	2C4A		2DCD		2DCD	CLR
	2C6F		2DF2		2DF2	Copy Cursor ausschalten
	2C76		2DFA		2DFA	beide Cursor vergleichen
	2C82		2E06		2E06	neue Copy Cursor-Zeile berechnen
	2C98		2E17		2E17	SHIFT-CUR RGHT
	2C9D		2E1C		2E1C	SHIFT-CUR LEFT
	2CA2		2E21		2E21	SHIFT-CUR UP
	2CA7		2E26		2E26	SHIFT-CUR DOWN
	2CAA		2E29		2E29	Copy Cursor bewegen
	2CCD		2E4A		2E4A	Copy Cursor anschalten
	2CD2		2E4F		2E4F	Copy Cursor ausschalten
	2CD5		2E52		2E52	Copy Cursor umschalten
	2CEA		2E65		2E65	COPY
	2D29		2EA2		2EA2	Copy Cursor nach rechts
	2D2D		2EA6		2EA6	Copy Cursor nach links
	2D2F		2EA8		2EA8	Copy Cursor verschieben
			2EC1		2EC1	Copy Cursor ausgeschaltet?
	2D4A		2EC7		2EC7	Cursor nach links
	2D50		2ECD		2ECD	Cursor nach rechts
	2D54		2ED1		2ED1	Cursor verschieben
	2D67		2EE4		2EE4	Buffer ab HL ausgeben
	2D85		2F02		2F02	Zeichen ausgeben
	2DA8		2F25		2F25	Zeichen ausgeben
	2DD9		2F56		2F56	Zeichen von Tastatur holen
	2DF6					Adresse aus Tabelle holen
BD3D	2E18	BD5E	2F91	BD61	2F91	FLO Zahl kopieren
BD40	2E29	BD61	2F9F	BD64	2F9F	FLO INT nach REAL

BD43	2E55	BD64	2FC8	BD67	2FC8	FLO 4-Bytes nach REAL
BD94	2E5E	BDB5	2FD1	BDB8	2FD1	FLO 5-Bytes nach REAL
BD46	2E66	BD67	2FD9	BD6A	2FD9	FLO REAL nach INTEGER
BD49	2E8E	BD6A	3001	BD6D	3001	FLO Zahl runden
BD4C	2EA1	BD6D	3014	BD70	3014	FLO FIX-Funktion
BD4F	2EAC	BD70	3055	BD73	3055	FLO INT-Funktion
BD52	2EB6	BD73	305F	BD76	305F	FLO Zahl für Dezimalwandlung aufbereiten
BD55	2F1D	BD76	30C6	BD79	30C6	FLO Zahl mit 10^A multiplizieren
	2F3E					Restexponent und Zehnerpotenz holen
BD97	2F94	BDB8	3136	BDBB	3136	FLO RND INITIALIZE
BD9A	2FA1	BDBB	3143	BDBE	3143	FLO RND SEED
BD9D	2FB7	BD7C	3159	BD7F	3159	FLO RND-Funktion
BDA0	2FE6	BD88	3188	BD8B	3188	FLO letzter RND-Wert
	2FFA		319C		319C	neues RND-Word generieren
BD82	300F	BDA3	31B1	BDA6	31B1	FLO LOG10-Funktion
BD7F	3014	BDA0	31B6	BDA3	31B6	FLO LOG-Funktion
	3017		31B9		31B9	FLO LOGARITHMUS
	304F		31EE		31EE	Konstanten für LOGARITHMUS
	3081		3220		3220	diverse FLO-Konstanten
BD85	3090	BDA6	322F	BDA9	322F	FLO EXP-Funktion
	30C1		3260		3260	Konstanten für EXP, 1. Polynom
	30D5		3274		3274	Konstanten für EXP, 2. Polynom
	30FB		329D		329D	diverse Konstanten für EXP
BD79	310A	BD9A	32AC	BD9D	32AC	FLO SQR-Funktion
BD7C	310D	BD9D	32AF	BDA0	32AF	FLO Potenzierung
	3182		3324		3324	Vorzeichen des Ergebnisses bestimmen
BD76	31A3	BD97	2F73	BD9A	2F73	FLO PI-Funktion
	31A9		2F78		2F78	Konstante Pi
BD73	31AE	BD94	3345	BD97	3345	FLO DEG/RAD
BD8B	31B2	BDAC	3349	BDAF	3349	FLO COS-Funktion
BD88	31BC	BDA9	3353	BDAC	3353	FLO SIN-Funktion
	31BD		3354		3354	allgemeine SIN/COS-Funktion
	31EB		3382		3382	Konstanten für SIN/COS
	321D		33B4		33B4	diverse Konstanten für SIN/COS
BD8E	3231	BDAF	33C8	BDB2	33C8	FLO TAN-Funktion
BD91	3241	BDB2	33D8	BDB5	33D8	FLO ATN-Funktion
	3257		33EE		33EE	Konstanten für ATN
	32A9		3440		3440	Polynomberechnung mit Arg^2
	32AC		3443		3443	Polynomberechnung
	32D4		3469		3469	Argument normalisieren für EXP
	32D5		346A		346A	Argument normalisieren für SIN/COS
	32FD		35FB		35FB	FLO Kehrwert bilden
	3307		3492		3492	Exponenten vergleichen
	330F		2F8D		2F8D	Argument nach FAC1 kopieren
	3316		2F87		2F87	Argument nach FAC3 kopieren
	331D		3570		3570	Zahl quadrieren
	3328		2F7D		2F7D	Konstante 1 holen
	3332		2F82		2F82	Konstante 1
BD5B	3337		349A		349A	FLO (HL):=(HL)-(DE)
BD5E	333B	BD7F	349E	BD82	349E	FLO (HL):=(DE)-(HL)
BD58	333F	BD79	34A2	BD7C	34A2	FLO (HL):=(HL)+(DE)
	3340		34A3		34A3	allgemeine Addition/Subtraktion
	335A		34BC		34BC	Addieren
	3412					FLO (HL):=10*(HL)
BD61	3415	BD82	3577	BD85	3577	FLO (HL):=(HL)*(DE)
	3450		35B0		35B0	Ergebnismantisse berechnen
	346C		35CC		35CC	Byte mit FLO-Mantisse multiplizieren
	3493		35F3		35F3	Byte mit FLO-Mantisse multiplizieren

	349B					FLO (HL):=(HL)/10
BD64	349E	BD85	3604	BD88	3604	FLO (HL):=(HL)/(DE)
	3507		3672		3672	byteweise dividieren
	3532		369D		369D	Mantissenvergleich
	3548		36AF		36AF	Exponenten addieren, Vorzeichen berech.
	356C		36D3		36D3	Exponenten holen
BD67	3578					FLO (HL):=(HL)*2^A
BD6A	359A	BD8B	36DF	BD8E	36DF	FLO Vergleich (HL)-(DE)
BD70	35E8	BD91	3727	BD94	3727	FLO SGN-Funktion
BD6D	35F8	BD8E	3731	BD91	3731	Vorzeichen invertieren
	3604					FLO Nachkommastellen abschneiden
	3614		3029		3029	Nachkommastellen abschneiden
	363D		373D		373D	FLO Argument rechtsverschieben
	3673		3773		3773	FLO Zahl normalisieren
	36A1		379C		379C	FLO 4-Byte Mantisse nach Real
	36BA		37B7		37B7	Mantisse runden, Vorzeichen eintragen
	36E6		37E2		37E2	Unterlauf, Null setzen
	36EC		37E8		37E8	Überlauf, maximale positive Zahl setzen
	36EE					Überlauf, maximalen Betrag setzen

BDA3	3708		DD2F		DD2A	INT Zweierkomp.-Zahl f. Dezimalwandlung nach signed Binary, Wandlungs-Params
BDA6	370E		DD35		DD30	INT Dezimalwandlungs-Parameter für positive Integerzahl holen
BDA9	3715		DD3C		DD37	INT signed Binary nach Zweierkomplement
BDAC	3728		DD4F		DD4A	INT HL:=HL+DE
BDB2	3730		DD57		DD52	INT HL:=DE-HL
BDAF	3731		DD58		DD53	INT HL:=HL-DE
BDB5	3739		DD60		DD5B	INT HL:=HL*DE
	3745		DD6C		DD67	Vorzeichen des Ergebnisses bestimmen
BDBE	3750		DD77		DD72	vorzeichenlose Multiplikation
BDB8	377A		DDA1		DD9C	INT HL:=HL DIV DE
BDBB	3781		DDA8		DDA3	INT HL:=HL MOD DE
	3789		DDB0		DDAB	INT Division
BDC1	378C		DDB3		DDAE	vorzeichenlose Division
	37D1		DDEF		DDEA	INT HL:=ABS(HL)
BDC7	37D4		DDF2		DDED	INT HL:=-HL
BDCA	37E0		DDFE		DDF9	INT A:=SGN(HL)
BDC4	37E9		DE07		DE02	INT Vergleich HL-DE

5.4 Die Routinen des Basics

5.4.1 Die Routinen nach Adressen sortiert

464	664	6128	Routine
C000	C000	C000	Kennungen des Basics
C006	C006	C006	Basic-Kaltstart
C03F	C033	C033	Meldung des Basics
C04C	C040	C040	Name des ROMs
C052	C046	C046	Basic-Befehl EDIT
C064	C058	C058	Eingabeschleife
	C0AF	C0AF	normale Zeile holen/auswerten
C0CC	C0CD	C0CD	"Ready"
C0D3	C0DE	C0DE	AUTO ausschalten
C0D6	C0E1	C0E1	AUTO-Zeilennummer setzen
C0DF	C0EA	C0EA	Basic-Befehl AUTO
C102	C10D	C10D	AUTO-Eingabezeile holen
C12B	C128	C128	Basic-Befehl NEW
C132	C12F	C12F	Basic-Befehl CLEAR
	C13F	C13F	Basic-Befehl CLEAR INPUT
C13E	C145	C145	NEW Fortsetzung
C162	C166	C166	Ausdruckauswertung und I/O initialisieren
C16B	C16F	C16F	Basic initialisieren
C17A	C189	C189	Basic-Zeiger initialisieren
C18C	C178	C178	Variablen löschen
C19D	C1A4	C1A1	Ein-/Ausgabekanäle initialisieren
C1A2	C1A9	C1A6	neue Streamnummer setzen
C1AF	C136	C136	neue Eingabekanal-Nummer setzen
C1BA	C1C1	C1BE	aktuelle Streamnummer holen
C1C0	C1C7	C1C4	aktuelle Eingabekanalnummer holen
C1C6	C1CD	C1CA	optionale Streamnummer holen/setzen
C1CB	C1D2	C1CF	optionale Eingabekanalnummer holen/setzen
	C1D7	C1D4	optionale Eingabekanalnr. transparent setzen/rücksetzen
C1D0	C1E8	C1E5	optionale Streamnummer transparent setzen/rücksetzen
C1E3	C1FE	C1FB	optionale Filenummer holen
C1F5	C210	C20D	Filenummer holen
C1FB	C216	C213	Byte kleiner A holen
	C223	C220	Bytewert <2 (als Flag) holen
C20A	C23C	C239	Basic-Befehl PAPER
C212	C227	C224	Basic-Befehl PEN
C221	C24B	C248	Basic-Befehl BORDER
C22A	C254	C251	Basic-Befehl INK
C23C	C265	C262	2 Farbwerte holen
C24B	C274	C271	Farbstiftnummer holen
C24F	C278	C275	Basic-Befehl MODE
C25A	C283	C280	Basic-Befehl CLS
	C28C	C289	Window-Nummer transparent setzen
	C29B	C298	Basic-Funktion COPYCHR$
C262	C2A4	C2A1	Basic-Funktion VPOS
C267	C2CA	C2C7	Cursorzeile holen
C276	C2AD	C2AA	Basic-Funktion POS
C290			horizontale Position für I/O holen
C29F	C2D2	C2CF	aktuelle Ausgabe-Breite holen

```
C2B9 C2EA C2E7  auf Platz in Zeile prüfen
C2BF            auf Platz in Zeile prüfen
C2D2 C302 C2FF  Basic-Befehl LOCATE
C2E1 C311 C30E  Basic-Befehl WINDOW
C2FD C32B C328  Basic-Befehl WINDOW SWAP
C312 C341 C33E  Window-Nummer holen
C319 C346 C343  Basic-Befehl TAG
C320 C34D C34A  Basic-Befehl TAGOFF
C327 C354 C351  Koordinaten holen
     C363 C360  Basic-Befehl CURSOR
C337 C380 C37D  I/O initialisieren, String ausgeben
C341 C38E C38B  String ausgeben
C34E C39B C398  Linefeed ausgeben
C356 C3A3 C3A0  Zeichen ausgeben
C35C C3AB C3A8  Zeichen ausgeben
C377 C3C4 C3C1  Zeichen ausgeben (ohne LF-Behandlung)
C386 C3D3 C3D0  Bildschirm initialisieren
C392 C3E5 C3E2  Linefeed auf Bildschirm ausgeben
C39C C3EF C3EC  Cursorspalte holen
C3A8 C3F8 C3F5  Linefeed an Drucker ausgeben
C3B5 C3FF C3FC  Zeichen an Drucker ausgeben
C3DF            Druckkopfposition holen, nach A
C3E3 C42D C42A  Basic-Befehl WIDTH
C3EA C434 C431  Linefeed an Kassette ausgeben
C3F8 C43B C438  Zeichen an Kassette ausgeben
C414            Zeichen zurück in Kassettenbuffer
C417 C452 C44F  Basic-Funktion EOF
     C45F C45C  Zeichen von Kassette lesen
C424            Zeichen einlesen
C439 C472 C46F  Zeichen von Tastatur holen
C43C C475 C472  auf ESC-Taste prüfen
C453            ESC-Abbruch einmal ermöglichen
     C482 C47F  ESC-Abbruch ggf. einmal ermöglichen
C45E C495 C492  Break-Event-Routine
C46F C4A4 C4A1  nach ESC/Break auf Taste warten
     C4D3 C4D0  Basic-Befehl ON BREAK CONT
     C4D6 C4D3  ON BREAK CONT ausschalten
C48C C4E1 C4DE  Basic-Befehl ORIGIN
C4B5 C509 C506  Basic-Befehl CLG
     C515 C512  Basic-Befehl FILL
C4C6 C53C C539  Basic-Befehl DRAW
C4CB C541 C53E  Basic-Befehl DRAWR
C4D0 C546 C543  Basic-Befehl PLOT
C4D5 C54B C548  Basic-Befehl PLOTR
C4E9 C547 C544  Basic-Befehl TEST
C4EE C579 C576  Basic-Funktion TESTR
C505 C532 C52F  Basic-Befehl MOVE
C50A C537 C534  Basic-Befehl MOVER
C51A C58F C58C  Graphik-Koordinaten holen
     C59D C59A  Basic-Befehl GRAPHICS
     C5A1 C59E  Basic-Befehl GRAPHICS PEN
     C5B4 C5B1  Basic-Befehl GRAPHICS PAPER
     C5C3 C5C0  Basic-Befehl MASK
C529 C5D7 C5D4  Basic-Befehl FOR
C5FB C6A5 C6A2  Basic-Befehl NEXT
C632 C6DC C6D9  offene FOR-Schleife suchen
C661 C705 C702  Stepwert ggf. addieren, Ende prüfen
C6C7 C76A C767  Basic-Befehl IF
```

```
C6E8 C789 C786  Basic-Befehl GOTO
C6ED C78F C78C  Basic-Befehl GOSUB
C6F6 C796 C793  GOSUB-Datensatz auf Stack
C70F C7B3 C7B0  Basic-Befehl RETURN
C72E C7D2 C7CF  GOSUB auf Basic-Stack suchen
C747 C7EA C7E7  Basic-Befehl WHILE
C776 C81D C81A  Basic-Befehl WEND
C7B8 C860 C85D  offene WHILE-Schleife suchen
C7E3 C885 C882  Basic-Befehl ON
C807 C8B5 C8B2  Synchronous Events bearbeiten
C847 C8F5 C8F2  Break-Event Fortsetzung
C861 C914 C911  Zeilenadresse in Event-Block speichern
C879 C929 C926  Event-Routine für AFTER/EVERY/SQ
C8A4 C954 C951  RETURN Fortsetzung (AFTER/EVERY/SQ)
C8B6 C964 C961  RETURN Fortsetzung (ON BREAK)
C8CB C979 C976  Basic-Befehl ON BREAK
C8E1 C99A C997  Basic-Befehl DI
C8E7 C9A0 C99D  Basic-Befehl EI
C8ED C9A6 C9A3  Events für Basic initialisieren
C924 C9DD C9DA  Event-Block-Gruppe initialisieren
C940 C9F8 C9F5  Basic-Befehl ON SQ
C95D CA13 CA10  Adresse des SQ-Event-Blocks holen
C971 CA25 CA22  Basic-Befehl AFTER
C979 CA2D CA2A  Basic-Befehl EVERY
C99F CA53 CA50  Basic-Funktion REMAIN
C9B1 CA65 CA62  Event-Block-Adresse berechnen
C9C5 CA79 CA76  zugehöriges NEXT suchen
CA18 CACC CAC9  zugehöriges WEND suchen
     CAEF CAEC  Eingabezeile für LINE INPUT holen
CA3B CAFC CAF9  Eingabezeile holen
CA43 CB04 CB01  Buffer ausgeben, Zeile holen
CA4C CB0D CB0A  Zeile von Kassette holen
CA84 CB3A CB37  Fehlernummer und -zeile initialisieren
CA85 CB3B CB38  Fehlernummer setzen
     CB48 CB45  Fehler entsprechend Byte nach Aufruf melden
     CB4C CB49  Ausgabe von "Syntax error"
     CB50 CB4D  Ausgabe von "Improper argument"
CA8F CB54 CB51  Basic-Befehl ERROR
CA94 CB58 CB55  Fehler behandeln
CADF CBAD CBAA  Error-Zeilennummer holen
CAEA CBB8 CBB5  "Division by zero" ausgeben
CAF3 CBC1 CBBE  "Overflow" ausgeben
CB18 CBE9 CBE6  "Undefined line xxxxx in yyyyy" ausgeben
CB23 CBF4 CBF1  "Undefined line"
CB33 CC04 CC01  "Break in" Zeilennummer ausgeben
CB36 CC07 CC04  Meldung mit Zeilennummer ausgeben
CB4F CC1F CC1C  "Break", " in "
CB5A CC29 CC26  Basic-Befehl STOP
CB65 CC34 CC31  Basic-Befehl END
     CC3A CC37  DERR setzen, "Broken in" melden
CB6B CC41 CC3E  "Break" ausgeben, Abbruch behandeln
CB76 CC4C CC49  Programmende behandeln
CB93 CC69 CC66  PC und Zeilenadresse für CONT retten
CBAB CC81 CC7E  CONT sperren
CBB0 CC86 CC83  PC und Zeilenadresse für CONT retten
CBC0 CC96 CC93  Basic-Befehl CONT
CBD9 CCAE CCAB  ON ERROR ausschalten
CBE5 CCBB CCB8  Basic-Befehl ON ERROR
```

CBF8	CCCD	CCCA	Basic-Befehl ON ERROR GOTO 0
CC03	CCD8	CCD5	Basic-Befehl RESUME
CC19	CCEB	CCE8	RESUME ohne Parameter
CC20	CCF2	CCEF	RESUME NEXT
CC2B	CCFD	CCFA	Basic-Zeiger für RESUME setzen
CC45	CE8F	CE8C	Adresse des Fehlerstrings holen
CC5B	CD17	CD14	Tabelle der Fehlermeldungen
CE67	CEBB	CEB8	Byte-Ausdruck holen
CE6D	CEC6	CEC3	Byte-Ausdruck <>0 holen
CE7C	CED1	CECE	Integer von 0..32767 holen
CE86	CEDB	CED8	Integer von -32768..32767 holen
	CEE6	CEE3	Parameter für CALL/RSX holen
CE91	CEF8	CEF5	Integer von -32768..65535 holen
CE9F	CF06	CF03	String holen, vom Stringstack löschen
CEA5	CF0C	CF09	Stringausdruck holen
CEB0	CF12	CF0F	Zeilennummernbereich holen
CEE1	CF4B	CF48	Zeilennummer holen
CEFB	CF65	CF62	Ausdruck holen
CF07	CF70	CF6D	Teilausdruck holen
CF1E	CF88	CF85	Stringverknüpfung "+"
CF30	CF9A	CF97	Operator behandeln
CF59	CFC8	CFC5	Vergleichsoperator auswerten
CF81	CFF0	CFED	Tabelle der Hierarchiecodes und Operatorenadressen
CFAA	D011	D00E	numerischer Vergleich
CFB9	D020	D01D	"-" auswerten
CFC2	D02B	D028	NOT auswerten
CFCB	D036	D033	Einzeloperanden holen
CFF2			Tabelle für Operandenauswertung
D00D	D077	D074	Variablenwert holen
D02C	D095	D092	Konstantenwert holen
D070	D0D4	D0D1	Ausdruck und ")" holen
D080	D0DD	D0DA	Funktionsauswertung
D0AE	D105	D102	Funktion anspringen (Gruppen 1/3)
D0BB	D113	D110	Funktion anspringen (Gruppe 2)
D0CA			Funktionsadressen, Tokens $40-$48
	D11A	D117	Funktionsadressen, Tokens $40-$49
	D12E	D12B	Basic-Funktion DERR
D0DC	D133	D130	Basic-Funktion ERR
D0E5	D13C	D139	Basic-Funktion TIME
D0EE	D145	D142	ERL auswerten
D0F4	D14B	D148	Basic-Funktion HIMEM
D0FA	D151	D14E	Variablenadresse nach FAC ("@")
D107	D164	D161	Basic-Funktion XPOS
D10E	D16B	D168	Basic-Funktion YPOS
D117	D174	D171	Basic-Befehl DEF
D130	D18D	D18A	definierte Funktion auswerten
D190	D1E8	D1E5	Funktionsadressen, Tokens $71-$7F
D1AE	D206	D203	Funktionsadressen, Tokens $00-$1D
D1EA	D242	D23F	Basic-Funktion MIN
D1EE	D246	D243	Basic-Funktion MAX
D219	D26D	D26A	Basic-Funktion ROUND
D246	D299	D296	Basic-Befehl CAT
D256	D2AB	D2A8	Basic-Befehl OPENOUT
D25F	D2B7	D2B4	Basic-Befehl OPENIN
D26A	D2C1	D2BE	Eingabefile öffnen
D273	D2CA	D2C7	File öffnen
D285	D2DE	D2DB	File öffnen Fortsetzung
D298	D2F0	D2ED	Basic-Befehl CLOSEIN

```
D2A1 D2F8 D2F5  Basic-Befehl CLOSEOUT
D2AD D303 D300  Kassette/Diskette initialisieren
D2C0 D316 D313  Basic-Befehl SOUND
D30D D362 D35F  Bytewert für SOUND holen
D317 D36C D369  Bytewert kleiner 8 holen
D31E D373 D370  Basic-Befehl RELEASE
D329 D37E D37B  Basic-Funktion SQ
D341 D396 D393  Integer von -128..+127 holen
D34E D3A1 D39E  Basic-Befehl ENV
D367 D3BA D3B7  Parametergruppe für ENV holen
D385 D3D7 D3D4  Basic-Befehl ENT
D3AE D400 D3FD  Parametergruppe für ENT holen
D3D8 D428 D425  Parametergruppen für ENV/ENT holen
D3FF D44F D44C  Integerwert von 0..4095 holen
D409 D459 D456  Basic-Funktion INKEY
D423 D473 D470  Basic-Funktion JOY
D439 D489 D486  Basic-Befehl KEY
D456 D4A3 D4A0  Basic-Befehl KEY DEF
D494 D4DE D4DB  Basic-Befehl SPEED
D4C3 D508 D505  Basic-Befehl SPEED WRITE
D4DB D520 D51D  Basic-Funktion PI
D4E7 D52C D529  Basic-Befehl DEG
D4EB D530 D52D  Basic-Befehl RAD
D4EF D534 D531  Basic-Funktion SQR
D4F4 D539 D536  Basic-Operator ^
D50A D54F D54C  REAL-Funktion/-Operator ausführen
D520 D563 D560  Basic-Funktion EXP
D525 D568 D565  Basic-Funktion LOG10
D52A D56D D56A  Basic-Funktion LOG
D52F D572 D56F  Basic-Funktion SIN
D534 D577 D574  Basic-Funktion COS
D539 D57C D579  Basic-Funktion TAN
D53E D581 D57E  Basic-Funktion ATN
D543 D586 D583  "Random number seed ? "
D559 D59C D599  Basic-Befehl RANDOMIZE
D584 D5C4 D5C1  Basic-Funktion RND
D5AE D5ED D5EA  Variablenbereich freigeben
D5BE D5FD D5FA  verkettete Listen der Variablen löschen
D5C6 D605 D602  verkettete Listen der Felder löschen
D5D2            definierte Funktionen löschen
     D611 D60E  definierte Funktionen und Variablenoffsets löschen
D5D9 D61A D617  1. Offset der VL der Funktionen holen
D5DB D61C D619  1. Offset der VL der Variablen holen
D5EA D62A D627  1. Offset für VL der Felder holen
D5FC D63B D638  DEFREAL A-Z
D601 D640 D63D  DEF-Typflag in Tabelle
D614 D653 D650  Basic-Befehl DEFSTR
D618 D657 D654  Basic-Befehl DEFINT
D61C D65B D658  Basic-Befehl DEFREAL
D64F D68C D689  LET bzw. RSX-Wort auswerten
D654 D691 D68E  Basic-Befehl LET
D666 D6A2 D69F  FAC an Variable zuweisen
D67D D6B9 D6B6  Basic-Befehl DIM
D686 D6C2 D6BF  Variable holen, ggf. neu anlegen
D690 D6CC D6C9  Variable holen, nicht anlegen
D6A2 D6DE D6DB  FN-Eintrag suchen, ggf. anlegen
D6B3 D6EF D6EC  einfache Variable holen, ggf. neu anlegen
D6C8 D704 D701  Adresse aus Offset berechnen
```

D6D6	D712	D70F	Variable überlesen, Typ holen
D6DE	D71A	D717	Variable suchen
D708	D744	D740	Eintrag in VL suchen
D731	D769	D765	Variablennamen überlesen
D73D	D773	D76F	FN-Eintrag neu anlegen
D749	D77F	D77B	einfache Variable neu anlegen
D76D	D7A2	D79E	Variablen-Offset ins Programm speichern
D777	D7AC	D7A8	Namenlänge holen, Platz berechnen
D78A	D7BC	D7B8	Namen und Typ übertragen
D7A5	D7D4	D7D0	Eintrag in VL einhängen
D7B5	D7E4	D7E0	eine Variable dimensionieren
D7DB	D80A	D806	Variablenadresse holen, auf Feld prüfen
D85A	D887	D883	Indizes holen, auf Basic-Stack
D88A	D8B7	D8B3	Feldvariable neu anlegen
	D92B	D927	Word vom Basic-Stack holen
D906	D935	D931	Variablenname und Offset holen
D92B	D962	D95E	Variablennamen vom Basic-Stack
D939	D970	D96C	Variablennamen auf Basic-Stack
D97F	D9B3	D9AF	Variablentyp entsprechend Token setzen
D999	D9CD	D9C9	VL der Felder neu generieren
D9C0	D9F4	D9F0	Basic-Befehl ERASE
D9CC	DA00	D9FC	ein Feld löschen
D9FD	DA24	DA20	FN-Listenzeiger löschen
DA07	DA2E	DA2A	neuen Eintrag in FN-Liste generieren
DA27	DA4D	DA49	Eintrag in FN-Liste einhängen
DA30	DA56	DA52	Eintrag aus FN-Liste aushängen
DA4B	DA6E	DA6A	Funktionsvariable holen, in VL eintragen
DA74	DA97	DA93	sämtliche Stringvariablen durchgehen
DACE	DAED	DAE9	VL durchgehen, Routine ausführen
DAE7	DB06	DB02	Stringbearbeitungsroutine ausführen
DAF8	DB18	DB13	Basic-Befehl LINE INPUT
DB1A	DB36	DB31	Zeile für LINE INPUT holen
DB2B	DB48	DB43	Basic-Befehl INPUT
DB47	DB60	DB5B	Eingabezeile holen und prüfen
DB77	DB7E	DB79	"?Redo from start"
DB89	DB90	DB8B	ggf. Text ausgeben, Flags holen
DBAD			Eingabezeile von Tastatur holen
	DBB6	DBB1	Text für INPUT holen und ausgeben
DBBC	DBC2	DBBD	Eingabe an Variable zuweisen
DBD3	DBD2	DBCD	Eingabezeile überprüfen
DC02	DBFC	DBF7	Eingabe auswerten
DC21	DC1A	DC15	Eingabestring holen
DC38	DC31	DC2C	numerische Eingabe (von Kassette/Diskette)
DC47	DC3D	DC38	Eingabestring holen (von Kassette/Diskette)
DC66	DC5C	DC57	Eingabezeile (von Kassette/Diskette) holen
DC6E	DC64	DC5F	Eingabe bis Trennzeichen holen
DC9D	DC93	DC8E	Zeichen holen, " ", TAB, LF überlesen
DCA8	DC9E	DC99	Zeichen holen, CR/LF auswerten
DCC6	DCBA	DCB5	auf Space, TAB, LF, Komma, CR prüfen
DCD0	DCC4	DCBF	auf Space, TAB, LF prüfen
DCD9	DCCD	DCC8	Basic-Befehl RESTORE
DCEB	DCDF	DCDA	Basic-Befehl READ
DD17	DD0F	DD0A	nächstes DATA-Element suchen
	DD2F	DD2A	Integer-Arithmetik (siehe 5.3)
	DE1A	DE15	Test auf Komma
	DE1E	DE19	Test auf Klammer auf
	DE22	DE1D	Test auf Klammer zu
	DE26	DE21	Test auf "="

DD37	DE2A	DE25	Test auf Zeichen nach Aufruf
DD3F	DE31	DE2C	nächstes Zeichen holen
DD4A	DE3C	DE37	auf Statementende prüfen
DD51	DE4C	DE47	Test auf Statementende
DD55	DE46	DE41	Test auf Komma
DD61	DE52	DE4D	Spaces, TABs und LFs überlesen
DD71	DE62	DE5D	Statement nochmals ausführen
DD74	DE65	DE60	Interpreterschleife
DDAB	DE94	DE8F	Befehl ausführen
DDCB	DEAF	DEAA	Direkt-Modus einschalten
DDD2	DEB6	DEB1	Zeilenadresse nach HL holen
DDD6	DEBA	DEB5	Zeilennummer/Direkt-Modus-Flag holen
DDE2	DEC6	DEC1	Basic-Befehl TRON
DDE6	DECA	DEC5	Basic-Befehl TROFF
DDEB	DECF	DECA	Trace-Routine
DE01	DEE5	DEE0	Adressen der Basic-Befehle
DEBB	DFA9	DFA4	Zeile tokenisieren
DEE1	DFCD	DFC8	ein Item tokenisieren
DF09	DFF1	DFEC	Buchstaben auswerten
DF25	E00D	E008	Zeichen in Token-Buffer
DF30	E017	E012	Tabelle der Tokens mit Sonderteil
DF35	E01C	E017	Zeile bis Statementende übernehmen
DF4E	E03F	E03A	Keyword/Variable tokenisieren
DF89	E075	E070	Variablennamen auswerten
DFC8	E0B4	E0AF	Befehlstoken behandeln
DFDC	E0CB	E0C6	Tabelle der Tokens mit Zeilennummer
DFEA	E0D6	E0D1	Variablentyp feststellen
DFFF	E0E7	E0E2	Dezimalzahl tokenisieren
E04A	E12B	E126	Eingabe bis DE übernehmen
E05A	E13B	E136	Hex/Binärzahl tokenisieren
E080	E161	E15C	Sonderzeichen auswerten
E0B3	E18E	E189	Flag für Variable/Zeilennummer prüfen
E0BF	E19A	E195	String in Buffer übernehmen
E0CD	E1A8	E1A3	RSX-Code auswerten
E0DF	E1BA	E1B5	Kennzeichen für Namen setzen
E0E6	E1C1	E1BC	"'" auswerten
E0F0	E1CB	E1C6	restliche Zeile übernehmen
E0F7	E1D2	E1CD	Basic-Befehl LIST
E10D	E1E8	E1E3	Programmbereich listen
E145	E222	E21D	Zeichen für LIST ausgeben
	E23D	E238	Zeile/Zeilennummer für AUTO nach ASCII wandeln
E163	E259	E254	Basic-Zeile nach ASCII wandeln
E196	E26B	E266	Item nach ASCII wandeln
E1DE	E2AF	E2AA	Konstante auswerten
E1E7	E2B8	E2B3	Variable auswerten
E1FE	E2CF	E2CA	Zeichen in LIST-Buffer
E205	E2D6	E2D1	RSX-Code auswerten
E20F	E2E0	E2DB	Namen übertragen
E21A	E2EB	E2E6	ggf. Space ausgeben
	E2F1	E2EC	REM-Token nach ASCII wandeln
E220	E2FD	E2F8	Keyword-Token nach ASCII wandeln
E253	E334	E32F	Konstante nach ASCII wandeln
E2DD	E3AD	E3A8	Zeiger in Keyword-Tabelle holen
E2ED	E3BD	E3B8	Token suchen, Keywordadresse holen
E313	E3DC	E3D7	Token in Tabelle suchen
E327	E3F0	E3EB	String in Keyword-Tabelle suchen
E354	E41D	E418	Adressen der Keyword-Tabellen
E388	E451	E44C	Basic-Keyword-Tabellen

E64B	E73B	E736	Tabelle der Keywords ohne Buchstaben
E676	E766	E761	Programm löschen
E687	E775	E770	Zeilenadressen im Programm eliminieren
E69D	E78B	E786	Zeilenadressen im Statement durch Zeilennrn. ersetzen
E6BC			Eingabezeile auswerten
E6D2	E7AA	E7A5	Zeile im Programm einfügen
E70B	E7E9	E7E4	Bereich aus Programm löschen
E728	E7F3	E7EE	Basic-Befehl DELETE
E737	E805	E800	Löschbereich für DELETE holen
E75A	E81F	E81A	Programmbereich für DELETE löschen
E767	E82C	E827	Zeilenadresse holen
E79A	E861	E85C	Zeile suchen, ggf. Fehler ausgeben
E7A3	E869	E864	Zeile im Programm suchen
E7C1	E887	E882	nächsthöhere Zeile suchen
E7DF	E8A3	E89E	Basic-Befehl RENUM
E864	E925	E920	Zeilennummer im Statement ersetzen
E888	E949	E944	bei Zeilennummer im Statement Fehler ausgeben
E89F	E960	E95B	zugehöriges ELSE suchen
E8C1	E97F	E97A	Arrayindizes ggf. überlesen
E8EF	E9A8	E9A3	Basic-Befehl DATA
E8F3	E9AC	E9A7	Basic-Befehl REM
E8F3	E9B2	E9AD	Basic-Befehl ELSE
E8F3	E9B2	E9AD	Basic-Befehl '
E8FF	E9BE	E9B9	Programm durchgehen, Routine ausführen
E923	E9E2	E9DD	nächstes Statement suchen, Fehler bei Programmende
E935	E9F4	E9EF	Statementende/THEN/ELSE suchen
E943	EA02	E9FD	nächstes Item suchen
E95C	EA23	EA1E	String überlesen
E968	EA2F	EA2A	Variable überlesen
E978	EA39	EA34	Konstante überlesen
	EA4A	EA45	REM bzw. "'" überlesen
E989	EA52	EA4D	Variablenoffsets löschen
E996	EA5F	EA5A	Offsets im Statement löschen
E9BD	EA7D	EA78	Basic-Befehl RUN
E9F6	EABA	EAB5	Basic-Befehl LOAD
EA0D	EAD6	EAD1	1. Block des Programms lesen/auswerten
EA30	EAF6	EAF1	Binärdatei laden
EA3C	EB02	EAFD	Basic-Befehl CHAIN
EAA6	EB59	EB54	Basic-Befehl MERGE
EAB5	EB68	EB63	Programm mergen
	EC01	EBFC	Programmzeichen einlesen
	EC0E	EC09	EOF melden
EB48	EC1E	EC19	Zeile aus altem Programm kopieren
EB5E	EC31	EC2C	Programmzeile von Kassette/Diskette laden
EB84	EC50	EC4B	Zwei-Byte-Wert von Kassette/Diskette laden
EB8F	EC59	EC54	1. Block des Programms lesen
EB9D	EC67	EC62	normales bzw. ASCII-Programm mergen
EBA8	EC72	EC6D	normales bzw. ASCII-Programm laden
EBEF	ECBB	ECB6	ASCII-Programm laden
EC09	ECE1	ECDC	Basic-Befehl SAVE
EC2C			Tabelle für SAVE
EC3D	ED11	ED0C	SAVE ,P
EC5C	ED30	ED2B	SAVE ,B
EC87	ED58	ED53	SAVE ,A
ECA3	ED74	ED6F	ASCII nach binär wandeln
ECBE	ED8F	ED8A	String in positive Binärzahl wandeln
ECC6	ED97	ED92	String in positive Binärzahl wandeln
ECCD	ED9E	ED99	Hex-/Binär-String nach Integer

ECDC	EDAD	EDA8	Dezimalstring nach Integer/REAL
ED44	EE14	EE0F	Vorzeichen im String bestimmen
ED53	EE23	EE1E	Ziffernstring nach unpacked BCD wandeln
ED77	EE47	EE42	dezimalen Exponenten holen/berechnen
EDC9	EE99	EE94	nächstes Zeichen aus Zahl holen
EDCE	EE9E	EE99	unpacked BCD nach Binär wandeln
EE04	EED4	EECF	Zeilennummern-String wandeln
EE1C	EEEC	EEE7	Hex-/Binärstring nach Integer wandeln
EE35	EF05	EF00	Dezimalstring nach Integer wandeln
EE61	EF31	EF2C	Ziffernwert berechnen
EE79	EF49	EF44	positive Integerzahl ausgeben
EE82	EF4F	EF4A	positive Integerzahl nach ASCII wandeln
EE8F	EF5F	EF5A	Zahl nach ASCII, kein positives Vorzeichen setzen
EE9D	EF6D	EF68	Zahl nach ASCII, maximal 9 Ziffern
EE9F	EF6F	EF6A	Zahl formatiert nach ASCII wandeln
EED4	EF9B	EF96	Dezimalpunkt und Exponenten setzen
EEE4	EFAF	EFAA	normale Exponentialdarstellung
EF01	EFCC	EFC7	normale Darstellung
EF0A	EFD5	EFD0	Zahl mit Nachkommastellen
EF20	EFEB	EFE6	Exponentialdarstellung mit maximal 7 Mantissenstellen
EF27	EFF2	EFED	formatierte Exponentialdarstellung
EF88	F05B	F056	formatierte Darstellung
EFA0	F079	F074	Dezimalpunkt einfügen
EFB4	F08C	F087	führende Nullen in Buffer
EFC8	F09F	F09A	Zahl runden
EFE1	F0B9	F0B4	Zahl bei letzter Stelle um 1 erhöhen
EFEF	F0C7	F0C2	Nullen an Zahl anhängen
F00E	F0E3	F0DE	Nachkomma-Nullen unterdrücken
F025	F100	F0FB	Vorkommastellenzahl ohne Sonderzeichenstellen holen
F036	F118	F113	Nachkommastellen (ohne ".") holen
F03D	F11F	F11A	Komma-Einteilungen setzen
F050	F131	F12C	ggf. führende Null in Buffer
F05F	F14A	F145	ggf. führendes "$" in Buffer
	F14A	F145	ggf. führendes Währungszeichen setzen
F069	F154	F14F	Vorzeichen setzen
F06F			Zeichen ans Bufferende schreiben
F07C	F1F4	F1EF	führende Zeichen vor die Zahl setzen
F096	F18C	F187	Flag für Formatüberlauf setzen
F09B			Vorzeichenflags holen
F0B7	F18F	F18A	Binärzahl nach ASCII-Mantisse wandeln
F0DD			Binärzahl nach BCD wandeln
	F1C6	F1C1	gepackte BCD-Zahl nach ASCII wandeln
F114			Zahl nach Binärstring wandeln
F119			Zahl nach Hex-String wandeln
	F1E4	F1DF	Zahl nach Hex-/Binär-String wandeln
F158	F20D	F208	Basic-Funktion PEEK
F15F	F214	F20F	Basic-Befehl POKE
F16D	F21E	F219	Basic-Funktion INP
F177	F228	F223	Basic-Befehl OUT
F17D	F22E	F229	Basic-Befehl WAIT
F194	F23F	F23A	Adresse und Byte holen
F1A0	F24A	F245	RSX-Wort auswerten
F1BA	F261	F25C	Basic-Befehl CALL
F1BF	F266	F261	Parameter holen, Routine ausführen
F1F2	F29E	F299	ZONE-Default setzen
F1F6	F2A2	F29D	Basic-Befehl ZONE
F1FD	F2A9	F2A4	Basic-Befehl PRINT
F208			PRINT Fortsetzung

```
F224 F2C8 F2C3  Tabelle für PRINT
F233 F2D7 F2D2  PRINT, Ausdruck ausgeben
F25C F31E F319  PRINT, Komma-Tabulator
F277 F339 F334  PRINT SPC
F280 F342 F33D  PRINT TAB
F295 F357 F352  Spaces ausgeben
F2A0 F362 F35D  Integer in Klammern holen
F2AF F36E F369  Zahl der Spaces MOD Ausgabebreite
F2C4 F383 F37E  PRINT USING
F324 F3D2 F3CD  Ausdruck formatiert ausgeben
F350 F3FC F3F7  ggf. String ausgeben
F378 F421 F41C  String für USING ausgeben
F3A3 F436 F431  numerischen Ausdruck ausgeben
F3BA F44D F448  Formatstring für numerischen Ausdruck auswerten
F41B F4B3 F4AE  Formatstring weiter auswerten
     F507 F502  auf Währungszeichen prüfen
F47B F50D F508  Basic-Befehl WRITE
F4C4 F544 F53F  RAM-Zeiger initialisieren
F4EF F570 F56B  Basic-Befehl MEMORY
     F58F F58A  auf Platz oberhalb HIMEM prüfen
F501 F5B0 F5AB  Test auf Platz für Binärdatei
     F5E5 F5E0  Test, ob Adresse im Bereich liegt
     F5F1 F5EC  Adresse mit HIMEM+1 vergleichen
F51D F5FD F5F8  Größe des Stringbereichs holen
F52C F60C F607  Programm/Variablen-Zeiger korrigieren
F53A F61F F61A  Arrayzeiger korrigieren
F549            Variablen in Stringbereich retten
     F62E F629  Variablenbereich schützen
F571            Variablen aus Stringbereich zurückholen
     F63E F640  Variablenbereich wieder ungeschützt
F58E F652 F64F  Basic-Stackpointer initialisieren
F5A0 F665 F662  Eintrag vom Basic-Stack holen
F5AC F671 F66E  Basic-Stackpointer neu setzen
F5B0 F675 F672  Platz auf Basic-Stack reservieren
F5CA F68F F68C  Stringbereich löschen
F5D1 F696 F693  Stringbereich-Platz reservieren
F5E6            Stringbereich erweitern
F5F8 F6BB F6B8  Platz für Programm/Variable schaffen
F618            auf Platz prüfen
     F6F1 F6E5  Bereich löschen
F628 F708 F6FC  Größe des freien Speicherplatzes holen
     F713 F707  höchste freie Adresse nach Programm holen
          F714  Zeiger auf freien Basic-Bereich holen
F632 F720 F720  Eingabebuffer belegen
F637 F725 F725  Ausgabebuffer belegen
     F72A F72A  Ein-/Ausgabebuffer reservieren
F66D F759 F759  Eingabebuffer freigeben
F671 F75D F75D  Ausgabebuffer freigeben
F675 F761 F761  Ein-/Ausgabebuffer ggf. freigeben
F69D F784 F784  Basic-Befehl SYMBOL
F6CD F7B1 F7B1  Basic-Befehl SYMBOL AFTER
F706 F7E9 F7E9  User-Matrix neu setzen
F72E            Speicherbereich freigeben
F743            Platz im Speicher reservieren
F750 F808 F808  HIMEM neu setzen
F7BB F865 F865  Offset zu Stringadresse addieren
F7CB F879 F879  String überlesen, auf Stringstack
F7DC F88A F88A  String bis Zeilenende überlesen, auf Stringstack
```

F7E6	F894	F894	String bis Trennzeichen übernehmen
F7F9	F8A7	F8A7	Routine weiterführen, String auf Stringstack
F80F	F8B7	F8B7	Sonderzeichen am Stringende eliminieren
F828	F8D0	F8D0	String vom Stringstack löschen, ausgeben
	F8DC	F8DC	Teilstring ausgeben
F834	F8EC	F8EC	Basic-Funktion LOWER$
F839	F8F1	F8F1	auf Kleinschrift forcieren
F842	F8FA	F8FA	Basic-Funktion UPPER$
F863	F91D	F91D	Stringverknüpfung "+"
F88B			String kopieren
F897			Stringvergleich
	F959	F959	zwei Strings vom Stringstack
F8BA	F964	F964	Basic-Funktion BIN$
F8C4	F969	F969	Basic-Funktion HEX$
F8CE			Ausdruck und Stellenzahl holen
F8EA	F98F	F98F	Basic-Funktion DEC$
F91E	F9BC	F9BC	Basic-Funktion STR$
F93C	F9D3	F9D3	Basic-Funktion LEFT$
F943	F9D8	F9D8	Basic-Funktion RIGHT$
F94B	F9E2	F9E2	Basic-Funktion MID$
F971	F9F3	F9F3	Teilstring holen
F993	FA07	FA07	Basic-Befehl MID$
F9E9	FA43	FA43	String und Byte holen
F9FB	FA4F	FA4F	2. Byte für MID$ holen
FA0A	FA69	FA69	Basic-Funktion LEN
FA10	FA6E	FA6E	Basic-Funktion ASC
FA16	FA74	FA74	Basic-Funktion CHR$
FA24	FA7E	FA7E	Basic-Funktion INKEY$
FA2A			String für INKEY$ holen
FA36	FA8D	FA8D	Basic-Funktion STRING$
	FAA1	FAA1	FAC nach Byte/1. Stringzeichen wandeln
FA57	FAAD	FAAD	Basic-Funktion SPACE$
FA70			1. Zeichen aus String holen
FA77	FABE	FABE	Basic-Funktion VAL
FA92	FAD9	FAD9	FAC nach Byte wandeln
FAA1	FAE5	FAE5	Basic-Funktion INSTR
FAD4			Suchstring in String suchen
FB1B	FB4D	FB4D	Strings in Stringbereich forcieren
FB21	FB58	FB58	String in Stringbereich forcieren
FB2E	FB65	FB65	String in Stringbereich forcieren
FB49	FB8A	FB8A	Descriptor ggf. auf Stringstack
FB59	FB94	FB94	String kopieren, vom Stringstack löschen
FB8F	FBB9	FBB9	String in Stringbereich kopieren
FBA6			Stringdescriptor kopieren
FBB3	FBCC	FBCC	Stringdescriptorstack initialisieren
FBBA	FBD3	FBD3	Descriptor auf Stringstack und nach FAC
FBDA	FBF5	FBF5	String aus Stringbereich/Stringstack löschen
FBE8	FC03	FC03	String aus Stringbereich/Stringstack löschen
FBFF	FC1F	FC1F	Descriptor ggf. vom Stringstack löschen
	FC37	FC37	Test, ob Descriptor im Stringstack ist
FC19	FC41	FC41	Platz für String reservierenFC2D FC53 FC53 Basic-Funktion FRE
FC3E	FC64	FC64	Garbage collection
FC7B			höchste Stringadresse außerhalb Stringbereich suchen
FC9C			ggf. höchste Stringadresse setzen
	FCE3	FCE3	Descriptoradresse eintragen
FCB3	FCF3	FCF3	Parameter für Dezimalwandlung holen
FCC3	FD03	FD03	Dezimalwandlungsparams für positive Integerzahl holen
FCCC	FD0C	FD0C	Basic-Operator +

```
FCE1 FD21 FD21  Basic-Operator -
FCF5 FD35 FD35  Basic-Operator *
FD09 FD49 FD49  numerischer Vergleich
FD12 FD52 FD52  Basic-Operator /
FD37 FD67 FD67  Basic-Operator \
FD49 FD79 FD79  Basic-Operator MOD
FD58 FD87 FD87  Basic-Operator AND
FD63 FD92 FD92  Basic-Operator OR
FD6D FD9C FD9C  Basic-Operator XOR
FD77 FDA6 FDA6  Basic-Operator NOT
FD85 FDB0 FDB0  Basic-Funktion ABS
FDA3 FDC4 FDC4  Vorzeichen von FAC holen
FDAF FDD5 FDD5  Zahl runden, nach FAC
FDE8 FE0E FE0E  Basic-Funktion FIX
FDED FE13 FE13  Basic-Funktion INT
FE06 FE2C FE2C  Integer mit Vorzeichen nach Integer
FE15 FE3B FE3B  Typen angleichen, Werte holen
FE4F FE70 FE70  Integeroperanden nach REAL wandeln
FE60 FE89 FE89  positive Integerzahl nach REAL wandeln
FE63 FE8C FE8C  Integer nach REAL wandeln
FE6A FE93 FE93  Integer nach REAL wandeln
FE7C FEA5 FEA5  4-Byte-Integer nach REAL wandeln
FE8D FEB6 FEB6  Basic-Funktion CINT
FE93 FEBC FEBC  REAL im FAC nach Integer im FAC wandeln
FE9A FEC3 FEC3  Operanden nach Integer wandeln
FEA5 FECE FECE  FAC nach Integer wandeln
FEC2 FEEB FEEB  Basic-Funktion UNT
FED7 FEFF FEFF  FAC-Typ angleichen
FEE5 FF0D FF0D  FAC-Typ angleichen
FEEC FF14 FF14  Basic-Funktion CREAL
FEF3 FF1B FF1B  FAC löschen (FAC=0)
FF02 FF2A FF2A  Basic-Funktion SGN
FF05 FF2D FF2D  Zweierkomplements-Byte in A nach Integer in FAC
FF0A FF32 FF32  positives Byte in A nach Integer in FAC
FF0D FF35 FF35  Integer in HL nach FAC
FF16 FF3E FF3E  FAC auf REAL, Zeiger nach HL
FF1D FF45 FF45  Zeiger auf FAC und Typ holen
FF23 FF4B FF4B  Typ des FAC nach A holen
FF27 FF66 FF66  Typ des FAC holen, Flags setzen
FF2D FF4F FF4F  numerischen Wert aus FAC holen
FF3C FF5E FF5E  Test auf String, sonst Fehler
FF45 FF66 FF66  Typ des FAC holen, Flags setzen
FF4B FF6C FF6C  Wert nach FAC kopieren
FF53 FF74 FF74  FAC auf Basic-Stack
FF62 FF83 FF83  FAC kopieren
FF71 FF92 FF92  Test auf Buchstabe
FF7B FF9C FF9C  Test auf Buchstabe, Ziffer, "."
FF7F FFA0 FFA0  Test auf Ziffer oder Dezimalpunkt
FF83 FFA4 FFA4  Test auf Ziffer
FF8A FFAB FFAB  auf Großschrift forcieren
FF93 FFB4 FFB4  Adresse aus Tabelle entsprechend Zeichen holen
FFAA FFCA FFCA  Byte in Tabelle suchen
FFB8 FFD8 FFD8  HL und DE vergleichen
FFBE FFDE FFDE  HL und BC vergleichen
FFC4            DE:=HL-DE
FFCF            HL:=HL-DE
FFDA FFE4 FFE4  BC:=HL-DE
FFE7            HL:=HL-BC
```

	FFEC	FFEC	Block nach unten verschieben, Länge in A
FFF2	FFEF	FFEF	Block nach unten verschieben, Länge in BC
FFF5	FFF5	FFF5	Block nach oben verschieben, Länge in BC
FFF8	FFFB	FFFB	JP(HL)
FFF9	FFFC	FFFC	JP(BC)
FFFB	FFFE	FFFE	JP(DE)

5.4.2 Die Routinen alphabetisch sortiert

464	664	6128	Routine
E0E6	E1C1	E1BC	"'" auswerten
CFB9	D020	D01D	"-" auswerten
DB77	DB7E	DB79	"?Redo from start"
CB33	CC04	CC01	"Break in" Zeilennummer ausgeben
CB6B	CC41	CC3E	"Break" ausgeben, Abbruch behandeln
CB4F	CC1F	CC1C	"Break", " in "
CAEA	CBB8	CBB5	"Division by zero" ausgeben
CAF3	CBC1	CBBE	"Overflow" ausgeben
D543	D586	D583	"Random number seed ? "
C0CC	C0CD	C0CD	"Ready"
CB23	CBF4	CBF1	"Undefined line"
CB18	CBE9	CBE6	"Undefined line xxxxx in yyyyy" ausgeben
EB8F	EC59	EC54	1. Block des Programms lesen
EA0D	EAD6	EAD1	1. Block des Programms lesen/auswerten
D5D9	D61A	D617	1. Offset der VL der Funktionen holen
D5DB	D61C	D619	1. Offset der VL der Variablen holen
D5EA	D62A	D627	1. Offset für VL der Felder holen
FA70			1. Zeichen aus String holen
C23C	C265	C262	2 Farbwerte holen
F9FB	FA4F	FA4F	2. Byte für MID$ holen
FE7C	FEA5	FEA5	4-Byte-Integer nach REAL wandeln
D6C8	D704	D701	Adresse aus Offset berechnen
FF93	FFB4	FFB4	Adresse aus Tabelle entsprechend Zeichen holen
CC45	CE8F	CE8C	Adresse des Fehlerstrings holen
C95D	CA13	CA10	Adresse des SQ-Event-Blocks holen
	F5F1	F5EC	Adresse mit HIMEM+1 vergleichen
F194	F23F	F23A	Adresse und Byte holen
DE01	DEE5	DEE0	Adressen der Basic-Befehle
E354	E41D	E418	Adressen der Keyword-Tabellen
C29F	C2D2	C2CF	aktuelle Ausgabe-Breite holen
C1C0	C1C7	C1C4	aktuelle Eingabekanalnummer holen
C1BA	C1C1	C1BE	aktuelle Streamnummer holen
E8C1	E97F	E97A	Arrayindizes ggf. überlesen
F53A	F61F	F61A	Arrayzeiger korrigieren
ECA3	ED74	ED6F	ASCII nach binär wandeln
EBEF	ECBB	ECB6	ASCII-Programm laden
C43C	C475	C472	auf ESC-Taste prüfen
FF8A	FFAB	FFAB	auf Großschrift forcieren
F839	F8F1	F8F1	auf Kleinschrift forcieren
C2B9	C2EA	C2E7	auf Platz in Zeile prüfen
C2BF			auf Platz in Zeile prüfen
	F58F	F58A	auf Platz oberhalb HIMEM prüfen
F618			auf Platz prüfen
DCD0	DCC4	DCBF	auf Space, TAB, LF prüfen
DCC6	DCBA	DCB5	auf Space, TAB, LF, Komma, CR prüfen

DD4A	DE3C	DE37	auf Statementende prüfen
	F507	F502	auf Währungszeichen prüfen
F324	F3D2	F3CD	Ausdruck formatiert ausgeben
CEFB	CF65	CF62	Ausdruck holen
D070	D0D4	D0D1	Ausdruck und ")" holen
F8CE			Ausdruck und Stellenzahl holen
C162	C166	C166	Ausdruckauswertung und I/O initialisieren
	CB50	CB4D	Ausgabe von "Improper argument"
	CB4C	CB49	Ausgabe von "Syntax error"
F637	F725	F725	Ausgabebuffer belegen
F671	F75D	F75D	Ausgabebuffer freigeben
C0D3	C0DE	C0DE	AUTO ausschalten
C102	C10D	C10D	AUTO-Eingabezeile holen
C0D6	C0E1	C0E1	AUTO-Zeilennummer setzen
C16B	C16F	C16F	Basic initialisieren
E8F3	E9B2	E9AD	Basic-Befehl '
C971	CA25	CA22	Basic-Befehl AFTER
C0DF	C0EA	C0EA	Basic-Befehl AUTO
C221	C24B	C248	Basic-Befehl BORDER
F1BA	F261	F25C	Basic-Befehl CALL
D246	D299	D296	Basic-Befehl CAT
EA3C	EB02	EAFD	Basic-Befehl CHAIN
C132	C12F	C12F	Basic-Befehl CLEAR
	C13F	C13F	Basic-Befehl CLEAR INPUT
C4B5	C509	C506	Basic-Befehl CLG
D298	D2F0	D2ED	Basic-Befehl CLOSEIN
D2A1	D2F8	D2F5	Basic-Befehl CLOSEOUT
C25A	C283	C280	Basic-Befehl CLS
CBC0	CC96	CC93	Basic-Befehl CONT
	C363	C360	Basic-Befehl CURSOR
E8EF	E9A8	E9A3	Basic-Befehl DATA
D117	D174	D171	Basic-Befehl DEF
D618	D657	D654	Basic-Befehl DEFINT
D61C	D65B	D658	Basic-Befehl DEFREAL
D614	D653	D650	Basic-Befehl DEFSTR
D4E7	D52C	D529	Basic-Befehl DEG
E728	E7F3	E7EE	Basic-Befehl DELETE
C8E1	C99A	C997	Basic-Befehl DI
D67D	D6B9	D6B6	Basic-Befehl DIM
C4C6	C53C	C539	Basic-Befehl DRAW
C4CB	C541	C53E	Basic-Befehl DRAWR
C052	C046	C046	Basic-Befehl EDIT
C8E7	C9A0	C99D	Basic-Befehl EI
E8F3	E9B2	E9AD	Basic-Befehl ELSE
CB65	CC34	CC31	Basic-Befehl END
D385	D3D7	D3D4	Basic-Befehl ENT
D34E	D3A1	D39E	Basic-Befehl ENV
D9C0	D9F4	D9F0	Basic-Befehl ERASE
CA8F	CB54	CB51	Basic-Befehl ERROR
C979	CA2D	CA2A	Basic-Befehl EVERY
	C515	C512	Basic-Befehl FILL
C529	C5D7	C5D4	Basic-Befehl FOR
C6ED	C78F	C78C	Basic-Befehl GOSUB
C6E8	C789	C786	Basic-Befehl GOTO
	C59D	C59A	Basic-Befehl GRAPHICS
	C5B4	C5B1	Basic-Befehl GRAPHICS PAPER
	C5A1	C59E	Basic-Befehl GRAPHICS PEN
C6C7	C76A	C767	Basic-Befehl IF

```
C22A C254 C251  Basic-Befehl INK
DB2B DB48 DB43  Basic-Befehl INPUT
D439 D489 D486  Basic-Befehl KEY
D456 D4A3 D4A0  Basic-Befehl KEY DEF
D654 D691 D68E  Basic-Befehl LET
DAF8 DB18 DB13  Basic-Befehl LINE INPUT
E0F7 E1D2 E1CD  Basic-Befehl LIST
E9F6 EABA EAB5  Basic-Befehl LOAD
C2D2 C302 C2FF  Basic-Befehl LOCATE
     C5C3 C5C0  Basic-Befehl MASK
F4EF F570 F56B  Basic-Befehl MEMORY
EAA6 EB59 EB54  Basic-Befehl MERGE
F993 FA07 FA07  Basic-Befehl MID$
C24F C278 C275  Basic-Befehl MODE
C505 C532 C52F  Basic-Befehl MOVE
C50A C537 C534  Basic-Befehl MOVER
C12B C128 C128  Basic-Befehl NEW
C5FB C6A5 C6A2  Basic-Befehl NEXT
C7E3 C885 C882  Basic-Befehl ON
C8CB C979 C976  Basic-Befehl ON BREAK
     C4D3 C4D0  Basic-Befehl ON BREAK CONT
CBE5 CCBB CCB8  Basic-Befehl ON ERROR
CBF8 CCCD CCCA  Basic-Befehl ON ERROR GOTO 0
C940 C9F8 C9F5  Basic-Befehl ON SQ
D25F D2B7 D2B4  Basic-Befehl OPENIN
D256 D2AB D2A8  Basic-Befehl OPENOUT
C48C C4E1 C4DE  Basic-Befehl ORIGIN
F177 F228 F223  Basic-Befehl OUT
C20A C23C C239  Basic-Befehl PAPER
C212 C227 C224  Basic-Befehl PEN
C4D0 C546 C543  Basic-Befehl PLOT
C4D5 C54B C548  Basic-Befehl PLOTR
F15F F214 F20F  Basic-Befehl POKE
F1FD F2A9 F2A4  Basic-Befehl PRINT
D4EB D530 D52D  Basic-Befehl RAD
D559 D59C D599  Basic-Befehl RANDOMIZE
DCEB DCDF DCDA  Basic-Befehl READ
D31E D373 D370  Basic-Befehl RELEASE
E8F3 E9AC E9A7  Basic-Befehl REM
E7DF E8A3 E89E  Basic-Befehl RENUM
DCD9 DCCD DCC8  Basic-Befehl RESTORE
CC03 CCD8 CCD5  Basic-Befehl RESUME
C70F C7B3 C7B0  Basic-Befehl RETURN
E9BD EA7D EA78  Basic-Befehl RUN
EC09 ECE1 ECDC  Basic-Befehl SAVE
D2C0 D316 D313  Basic-Befehl SOUND
D494 D4DE D4DB  Basic-Befehl SPEED
D4C3 D508 D505  Basic-Befehl SPEED WRITE
CB5A CC29 CC26  Basic-Befehl STOP
F69D F784 F784  Basic-Befehl SYMBOL
F6CD F7B1 F7B1  Basic-Befehl SYMBOL AFTER
C319 C346 C343  Basic-Befehl TAG
C320 C34D C34A  Basic-Befehl TAGOFF
C4E9 C547 C544  Basic-Befehl TEST
DDE6 DECA DEC5  Basic-Befehl TROFF
DDE2 DEC6 DEC1  Basic-Befehl TRON
F17D F22E F229  Basic-Befehl WAIT
C776 C81D C81A  Basic-Befehl WEND
```

```
C747 C7EA C7E7  Basic-Befehl WHILE
C3E3 C42D C42A  Basic-Befehl WIDTH
C2E1 C311 C30E  Basic-Befehl WINDOW
C2FD C32B C328  Basic-Befehl WINDOW SWAP
F47B F50D F508  Basic-Befehl WRITE
F1F6 F2A2 F29D  Basic-Befehl ZONE
FD85 FDB0 FDB0  Basic-Funktion ABS
FA10 FA6E FA6E  Basic-Funktion ASC
D53E D581 D57E  Basic-Funktion ATN
F8BA F964 F964  Basic-Funktion BIN$
FA16 FA74 FA74  Basic-Funktion CHR$
FE8D FEB6 FEB6  Basic-Funktion CINT
     C29B C298  Basic-Funktion COPYCHR$
D534 D577 D574  Basic-Funktion COS
FEEC FF14 FF14  Basic-Funktion CREAL
F8EA F98F F98F  Basic-Funktion DEC$
     D12E D12B  Basic-Funktion DERR
C417 C452 C44F  Basic-Funktion EOF
D0DC D133 D130  Basic-Funktion ERR
D520 D563 D560  Basic-Funktion EXP
FDE8 FE0E FE0E  Basic-Funktion FIX
FC2D FC53 FC53  Basic-Funktion FRE
F8C4 F969 F969  Basic-Funktion HEX$
D0F4 D14B D148  Basic-Funktion HIMEM
D409 D459 D456  Basic-Funktion INKEY
FA24 FA7E FA7E  Basic-Funktion INKEY$
F16D F21E F219  Basic-Funktion INP
FAA1 FAE5 FAE5  Basic-Funktion INSTR
FDED FE13 FE13  Basic-Funktion INT
D423 D473 D470  Basic-Funktion JOY
F93C F9D3 F9D3  Basic-Funktion LEFT$
FA0A FA69 FA69  Basic-Funktion LEN
D52A D56D D56A  Basic-Funktion LOG
D525 D568 D565  Basic-Funktion LOG10
F834 F8EC F8EC  Basic-Funktion LOWER$
D1EE D246 D243  Basic-Funktion MAX
F94B F9E2 F9E2  Basic-Funktion MID$
D1EA D242 D23F  Basic-Funktion MIN
F158 F20D F208  Basic-Funktion PEEK
D4DB D520 D51D  Basic-Funktion PI
C276 C2AD C2AA  Basic-Funktion POS
C99F CA53 CA50  Basic-Funktion REMAIN
F943 F9D8 F9D8  Basic-Funktion RIGHT$
D584 D5C4 D5C1  Basic-Funktion RND
D219 D26D D26A  Basic-Funktion ROUND
FF02 FF2A FF2A  Basic-Funktion SGN
D52F D572 D56F  Basic-Funktion SIN
FA57 FAAD FAAD  Basic-Funktion SPACE$
D329 D37E D37B  Basic-Funktion SQ
D4EF D534 D531  Basic-Funktion SQR
F91E F9BC F9BC  Basic-Funktion STR$
FA36 FA8D FA8D  Basic-Funktion STRING$
D539 D57C D579  Basic-Funktion TAN
C4EE C579 C576  Basic-Funktion TESTR
D0E5 D13C D139  Basic-Funktion TIME
FEC2 FEEB FEEB  Basic-Funktion UNT
F842 F8FA F8FA  Basic-Funktion UPPER$
FA77 FABE FABE  Basic-Funktion VAL
```

C262	C2A4	C2A1	Basic-Funktion VPOS
D107	D164	D161	Basic-Funktion XPOS
D10E	D16B	D168	Basic-Funktion YPOS
C006	C006	C006	Basic-Kaltstart
E388	E451	E44C	Basic-Keyword-Tabellen
FCF5	FD35	FD35	Basic-Operator *
FCCC	FD0C	FD0C	Basic-Operator +
FCE1	FD21	FD21	Basic-Operator -
FD12	FD52	FD52	Basic-Operator /
FD58	FD87	FD87	Basic-Operator AND
FD49	FD79	FD79	Basic-Operator MOD
FD77	FDA6	FDA6	Basic-Operator NOT
FD63	FD92	FD92	Basic-Operator OR
FD6D	FD9C	FD9C	Basic-Operator XOR
FD37	FD67	FD67	Basic-Operator \
D4F4	D539	D536	Basic-Operator ^
F58E	F652	F64F	Basic-Stackpointer initialisieren
F5AC	F671	F66E	Basic-Stackpointer neu setzen
CC2B	CCFD	CCFA	Basic-Zeiger für RESUME setzen
C17A	C189	C189	Basic-Zeiger initialisieren
E163	E259	E254	Basic-Zeile nach ASCII wandeln
FFDA	FFE4	FFE4	BC:=HL-DEDDAB DE94 DE8F Befehl ausführen
DFC8	E0B4	E0AF	Befehlstoken behandeln
E888	E949	E944	bei Zeilennummer im Statement Fehler ausgeben
E70B	E7E9	E7E4	Bereich aus Programm löschen
	F6F1	F6E5	Bereich löschen
C386	C3D3	C3D0	Bildschirm initialisieren
EA30	EAF6	EAF1	Binärdatei laden
F0B7	F18F	F18A	Binärzahl nach ASCII-Mantisse wandeln
F0DD			Binärzahl nach BCD wandeln
FFF5	FFF5	FFF5	Block nach oben verschieben, Länge in BC
	FFEC	FFEC	Block nach unten verschieben, Länge in A
FFF2	FFEF	FFEF	Block nach unten verschieben, Länge in BC
C847	C8F5	C8F2	Break-Event Fortsetzung
C45E	C495	C492	Break-Event-Routine
DF09	DFF1	DFEC	Buchstaben auswerten
CA43	CB04	CB01	Buffer ausgeben, Zeile holen
FFAA	FFCA	FFCA	Byte in Tabelle suchen
C1FB	C216	C213	Byte kleiner A holen
CE6D	CEC6	CEC3	Byte-Ausdruck <>0 holen
CE67	CEBB	CEB8	Byte-Ausdruck holen
	C223	C220	Bytewert <2 (als Flag) holen
D30D	D362	D35F	Bytewert für SOUND holen
D317	D36C	D369	Bytewert kleiner B holen
CBAB	CC81	CC7E	CONT sperren
C39C	C3EF	C3EC	Cursorspalte holen
C267	C2CA	C2C7	Cursorzeile holen
FFC4			DE:=HL-DE
D601	D640	D63D	DEF-Typflag in Tabelle
D130	D18D	D18A	definierte Funktion auswerten
D5D2			definierte Funktionen löschen
	D611	D60E	definierte Funktionen und Variablenoffsets löschen
D5FC	D63B	D638	DEFREAL A-Z
	CC3A	CC37	DERR setzen, "Broken in" melden
FBBA	FBD3	FBD3	Descriptor auf Stringstack und nach FAC
FB49	FB8A	FB8A	Descriptor ggf. auf Stringstack
FBFF	FC1F	FC1F	Descriptor ggf. vom Stringstack löschen
	FCE3	FCE3	Descriptoradresse eintragen

ED77	EE47	EE42	dezimalen Exponenten holen/berechnen
EFA0	F079	F074	Dezimalpunkt einfügen
EED4	EF9B	EF96	Dezimalpunkt und Exponenten setzen
EE35	EF05	EF00	Dezimalstring nach Integer wandeln
ECDC	EDAD	EDA8	Dezimalstring nach Integer/REAL
FCC3	FD03	FD03	Dezimalwandlungsparams für positive Integerzahl holen
DFFF	E0E7	E0E2	Dezimalzahl tokenisieren
DDCB	DEAF	DEAA	Direkt-Modus einschalten
C3DF			Druckkopfposition holen, nach A
D9CC	DA00	D9FC	ein Feld löschen
DEE1	DFCD	DFC8	ein Item tokenisieren
F675	F761	F761	Ein-/Ausgabebuffer ggf. freigeben
	F72A	F72A	Ein-/Ausgabebuffer reservieren
C19D	C1A4	C1A1	Ein-/Ausgabekanäle initialisieren
D7B5	D7E4	D7E0	eine Variable dimensionieren
D6B3	D6EF	D6EC	einfache Variable holen, ggf. neu anlegen
D749	D77F	D77B	einfache Variable neu anlegen
DBBC	DBC2	DBBD	Eingabe an Variable zuweisen
DC02	DBFC	DBF7	Eingabe auswerten
E04A	E12B	E126	Eingabe bis DE übernehmen
DC6E	DC64	DC5F	Eingabe bis Trennzeichen holen
F632	F720	F720	Eingabebuffer belegen
F66D	F759	F759	Eingabebuffer freigeben
D26A	D2C1	D2BE	Eingabefile öffnen
C064	C058	C058	Eingabeschleife
DC21	DC1A	DC15	Eingabestring holen
DC47	DC3D	DC38	Eingabestring holen (von Kassette/Diskette)
DC66	DC5C	DC57	Eingabezeile (von Kassette/Diskette) holen
E6BC			Eingabezeile auswerten
	CAEF	CAEC	Eingabezeile für LINE INPUT holen
CA3B	CAFC	CAF9	Eingabezeile holen
DB47	DB60	DB5B	Eingabezeile holen und prüfen
DBAD			Eingabezeile von Tastatur holen
DBD3	DBD2	DBCD	Eingabezeile überprüfen
DA30	DA56	DA52	Eintrag aus FN-Liste aushängen
DA27	DA4D	DA49	Eintrag in FN-Liste einhängen
D7A5	D7D4	D7D0	Eintrag in VL einhängen
D708	D744	D740	Eintrag in VL suchen
F5A0	F665	F662	Eintrag vom Basic-Stack holen
CFCB	D036	D033	Einzeloperanden holen
	EC0E	EC09	EOF melden
D0EE	D145	D142	ERL auswerten
CADF	CBAD	CBAA	Error-Zeilennummer holen
C453			ESC-Abbruch einmal ermöglichen
	C482	C47F	ESC-Abbruch ggf. einmal ermöglichen
C9B1	CA65	CA62	Event-Block-Adresse berechnen
C924	C9DD	C9DA	Event-Block-Gruppe initialisieren
C879	C929	C926	Event-Routine für AFTER/EVERY/SQ
C8ED	C9A6	C9A3	Events für Basic initialisieren
EF20	EFEB	EFE6	Exponentialdarstellung mit maximal 7 Mantissenstellen
D666	D6A2	D69F	FAC an Variable zuweisen
FF53	FF74	FF74	FAC auf Basic-Stack
FF16	FF3E	FF3E	FAC auf REAL, Zeiger nach HL
FF62	FF83	FF83	FAC kopieren
FEF3	FF1B	FF1B	FAC löschen (FAC=0)
FA92	FAD9	FAD9	FAC nach Byte wandeln
	FAA1	FAA1	FAC nach Byte/1. Stringzeichen wandeln
FEA5	FECE	FECE	FAC nach Integer wandeln

FED7	FEFF	FEFF	FAC-Typ angleichen
FEE5	FF0D	FF0D	FAC-Typ angleichen
C24B	C274	C271	Farbstiftnummer holen
CA94	CB58	CB55	Fehler behandeln
	CB48	CB45	Fehler entsprechend Byte nach Aufruf melden
CA85	CB3B	CB38	Fehlernummer setzen
CA84	CB3A	CB37	Fehlernummer und -zeile initialisieren
D88A	D8B7	D8B3	Feldvariable neu anlegen
D273	D2CA	D2C7	File öffnen
D285	D2DE	D2DB	File öffnen Fortsetzung
C1F5	C210	C20D	Filenummer holen
F096	F18C	F187	Flag für Formatüberlauf setzen
E0B3	E18E	E189	Flag für Variable/Zeilennummer prüfen
D73D	D773	D76F	FN-Eintrag neu anlegen
D6A2	D6DE	D6DB	FN-Eintrag suchen, ggf. anlegen
D9FD	DA24	DA20	FN-Listenzeiger löschen
EF88	F05B	F056	formatierte Darstellung
EF27	EFF2	EFED	formatierte Exponentialdarstellung
F3BA	F44D	F448	Formatstring für numerischen Ausdruck auswerten
F41B	F4B3	F4AE	Formatstring weiter auswerten
D0BB	D113	D110	Funktion anspringen (Gruppe 2)
D0AE	D105	D102	Funktion anspringen (Gruppen 1/3)
D1AE	D206	D203	Funktionsadressen, Tokens $00-$1D
D0CA			Funktionsadressen, Tokens $40-$48
	D11A	D117	Funktionsadressen, Tokens $40-$49
D190	D1E8	D1E5	Funktionsadressen, Tokens $71-$7F
D080	D0DD	D0DA	Funktionsauswertung
DA4B	DA6E	DA6A	Funktionsvariable holen, in VL eintragen
EFB4	F08C	F087	führende Nullen in Buffer
F07C	F1F4	F1EF	führende Zeichen vor die Zahl setzen
FC3E	FC64	FC64	Garbage collection
	F1C6	F1C1	gepackte BCD-Zahl nach ASCII wandeln
F050	F131	F12C	ggf. führende Null in Buffer
F05F	F14A	F145	ggf. führendes "$" in Buffer
	F14A	F145	ggf. führendes Währungszeichen setzen
FC9C			ggf. höchste Stringadresse setzen
E21A	E2EB	E2E6	ggf. Space ausgeben
F350	F3FC	F3F7	ggf. String ausgeben
DB89	DB90	DB8B	ggf. Text ausgeben, Flags holen
C72E	C7D2	C7CF	GOSUB auf Basic-Stack suchen
C6F6	C796	C793	GOSUB-Datensatz auf Stack
C51A	C58F	C58C	Graphik-Koordinaten holen
F628	F708	F6FC	Größe des freien Speicherplatzes holen
F51D	F5FD	F5F8	Größe des Stringbereichs holen
ECCD	ED9E	ED99	Hex-/Binär-String nach Integer
EE1C	EEEC	EEE7	Hex-/Binärstring nach Integer wandeln
E05A	E13B	E136	Hex/Binärzahl tokenisieren
F750	F808	F808	HIMEM neu setzen
FFBE	FFDE	FFDE	HL und BC vergleichen
FFB8	FFD8	FFD8	HL und DE vergleichen
FFE7			HL:=HL-BC
FFCF			HL:=HL-DE
C290			horizontale Position für I/O holen
	F713	F707	höchste freie Adresse nach Programm holen
FC7B			höchste Stringadresse außerhalb Stringbereich suchen
C337	C380	C37D	I/O initialisieren, String ausgeben
D85A	D887	D883	Indizes holen, auf Basic-Stack
FF0D	FF35	FF35	Integer in HL nach FAC

```
F2A0 F362 F35D  Integer in Klammern holen
FE06 FE2C FE2C  Integer mit Vorzeichen nach Integer
FE63 FE8C FE8C  Integer nach REAL wandeln
FE6A FE93 FE93  Integer nach REAL wandeln
D341 D396 D393  Integer von -128..+127 holen
CE86 CEDB CED8  Integer von -32768..32767 holen
CE91 CEF8 CEF5  Integer von -32768..65535 holen
CE7C CED1 CECE  Integer von 0..32767 holen
     DD2F DD2A  Integer-Arithmetik (siehe 5.3)
FE4F FE70 FE70  Integeroperanden nach REAL wandeln
D3FF D44F D44C  Integerwert von 0..4095 holen
DD74 DE65 DE60  Interpreterschleife
E196 E26B E266  Item nach ASCII wandeln
FFF9 FFFC FFFC  JP(BC)
FFFB FFFE FFFE  JP(DE)
FFF8 FFFB FFFB  JP(HL)
D2AD D303 D300  Kassette/Diskette initialisieren
C000 C000 C000  Kennungen des Basics
E0DF E1BA E1B5  Kennzeichen für Namen setzen
E220 E2FD E2F8  Keyword-Token nach ASCII wandeln
DF4E E03F E03A  Keyword/Variable tokenisieren
F03D F11F F11A  Komma-Einteilungen setzen
E1DE E2AF E2AA  Konstante auswerten
E253 E334 E32F  Konstante nach ASCII wandeln
E978 EA39 EA34  Konstante überlesen
D02C D095 D092  Konstantenwert holen
C327 C354 C351  Koordinaten holen
D64F D68C D689  LET bzw. RSX-Wort auswerten
C3A8 C3F8 C3F5  Linefeed an Drucker ausgeben
C3EA C434 C431  Linefeed an Kassette ausgeben
C392 C3E5 C3E2  Linefeed auf Bildschirm ausgeben
C34E C39B C398  Linefeed ausgeben
E737 E805 E800  Löschbereich für DELETE holen
C03F C033 C033  Meldung des Basics
CB36 CC07 CC04  Meldung mit Zeilennummer ausgeben
C46F C4A4 C4A1  nach ESC/Break auf Taste warten
F00E F0E3 F0DE  Nachkomma-Nullen unterdrücken
F036 F118 F113  Nachkommastellen (ohne ".") holen
C04C C040 C040  Name des ROMs
D78A D7BC D7B8  Namen und Typ übertragen
E20F E2E0 E2DB  Namen übertragen
D777 D7AC D7A8  Namenlänge holen, Platz berechnen
C1AF C136 C136  neue Eingabekanal-Nummer setzen
C1A2 C1A9 C1A6  neue Streamnummer setzen
DA07 DA2E DA2A  neuen Eintrag in FN-Liste generieren
C13E C145 C145  NEW Fortsetzung
EF01 EFCC EFC7  normale Darstellung
EEE4 EFAF EFAA  normale Exponentialdarstellung
     C0AF C0AF  normale Zeile holen/auswerten
EBA8 EC72 EC6D  normales bzw. ASCII-Programm laden
EB9D EC67 EC62  normales bzw. ASCII-Programm mergen
CFC2 D02B D028  NOT auswerten
EFEF F0C7 F0C2  Nullen an Zahl anhängen
DC38 DC31 DC2C  numerische Eingabe (von Kassette/Diskette)
F3A3 F436 F431  numerischen Ausdruck ausgeben
FF2D FF4F FF4F  numerischen Wert aus FAC holen
CFAA D011 D00E  numerischer Vergleich
FD09 FD49 FD49  numerischer Vergleich
```

DD17	DD0F	DD0A	nächstes DATA-Element suchen
E943	EA02	E9FD	nächstes Item suchen
E923	E9E2	E9DD	nächstes Statement suchen, Fehler bei Programmende
EDC9	EE99	EE94	nächstes Zeichen aus Zahl holen
DD3F	DE31	DE2C	nächstes Zeichen holen
E7C1	E887	E882	nächsthöhere Zeile suchen
C632	C6DC	C6D9	offene FOR-Schleife suchen
C7B8	C860	C85D	offene WHILE-Schleife suchen
F7BB	F865	F865	Offset zu Stringadresse addieren
E996	EA5F	EA5A	Offsets im Statement löschen
	C4D6	C4D3	ON BREAK CONT ausschalten
CBD9	CCAE	CCAB	ON ERROR ausschalten
FE9A	FEC3	FEC3	Operanden nach Integer wandeln
CF30	CF9A	CF97	Operator behandeln
	C1D7	C1D4	optionale Eingabekanalnummer transparent setzen/rücksetzen
C1CB	C1D2	C1CF	optionale Eingabekanalnummer holen/setzen
C1E3	C1FE	C1FB	optionale Filenummer holen
C1C6	C1CD	C1CA	optionale Streamnummer holen/setzen
C1D0	C1E8	C1E5	optionale Streamnummer transparent setzen/rücksetzen
	CEE6	CEE3	Parameter für CALL/RSX holen
FCB3	FCF3	FCF3	Parameter für Dezimalwandlung holen
F1BF	F266	F261	Parameter holen, Routine ausführen
D3AE	D400	D3FD	Parametergruppe für ENT holen
D367	D3BA	D3B7	Parametergruppe für ENV holen
D3D8	D428	D425	Parametergruppen für ENV/ENT holen
CB93	CC69	CC66	PC und Zeilenadresse für CONT retten
CBB0	CC86	CC83	PC und Zeilenadresse für CONT retten
F5B0	F675	F672	Platz auf Basic-Stack reservieren
F5F8	F6BB	F6B8	Platz für Programm/Variable schaffen
FC19	FC41	FC41	Platz für String reservieren
F743			Platz im Speicher reservieren
EE79	EF49	EF44	positive Integerzahl ausgeben
EE82	EF4F	EF4A	positive Integerzahl nach ASCII wandeln
FE60	FE89	FE89	positive Integerzahl nach REAL wandeln
FF0A	FF32	FF32	positives Byte in A nach Integer in FAC
F208			PRINT Fortsetzung
F277	F339	F334	PRINT SPC
F280	F342	F33D	PRINT TAB
F2C4	F383	F37E	PRINT USING
F233	F2D7	F2D2	PRINT, Ausdruck ausgeben
F25C	F31E	F319	PRINT, Komma-Tabulator
E8FF	E9BE	E9B9	Programm durchgehen, Routine ausführen
E676	E766	E761	Programm löschen
EAB5	EB68	EB63	Programm mergen
F52C	F60C	F607	Programm/Variablen-Zeiger korrigieren
E75A	E81F	E81A	Programmbereich für DELETE löschen
E10D	E1E8	E1E3	Programmbereich listen
CB76	CC4C	CC49	Programmende behandeln
	EC01	EBFC	Programmzeichen einlesen
EB5E	EC31	EC2C	Programmzeile von Kassette/Diskette laden
F4C4	F544	F53F	RAM-Zeiger initialisieren
FE93	FEBC	FEBC	REAL im FAC nach Integer im FAC wandeln
D50A	D54F	D54C	REAL-Funktion/-Operator ausführen
	EA4A	EA45	REM bzw. "'" überlesen
	E2F1	E2EC	REM-Token nach ASCII wandeln
E0F0	E1CB	E1C6	restliche Zeile übernehmen
CC20	CCF2	CCEF	RESUME NEXT
CC19	CCEB	CCE8	RESUME ohne Parameter

```
C8A4 C954 C951  RETURN Fortsetzung (AFTER/EVERY/SQ)
C8B6 C964 C961  RETURN Fortsetzung (ON BREAK)
F7F9 F8A7 F8A7  Routine weiterführen, String auf Stringstack
E0CD E1A8 E1A3  RSX-Code auswerten
E205 E2D6 E2D1  RSX-Code auswerten
F1A0 F24A F245  RSX-Wort auswerten
EC87 ED58 ED53  SAVE ,A
EC5C ED30 ED2B  SAVE ,B
EC3D ED11 ED0C  SAVE ,P
F80F F8B7 F8B7  Sonderzeichen am Stringende eliminieren
E080 E161 E15C  Sonderzeichen auswerten
F295 F357 F352  Spaces ausgeben
DD61 DE52 DE4D  Spaces, TABs und LFs überlesen
F72E            Speicherbereich freigeben
DD71 DE62 DE5D  Statement nochmals ausführen
E935 E9F4 E9EF  Statementende/THEN/ELSE suchen
C661 C705 C702  Stepwert ggf. addieren, Ende prüfen
FBDA FBF5 FBF5  String aus Stringbereich/Stringstack löschen
FBE8 FC03 FC03  String aus Stringbereich/Stringstack löschen
C341 C38E C38B  String ausgeben
F7E6 F894 F894  String bis Trennzeichen übernehmen
F7DC F88A F88A  String bis Zeilenende überlesen, auf Stringstack
FA2A            String für INKEY$ holen
F378 F421 F41C  String für USING ausgeben
CE9F CF06 CF03  String holen, vom Stringstack löschen
E0BF E19A E195  String in Buffer übernehmen
E327 E3F0 E3EB  String in Keyword-Tabelle suchen
ECBE ED8F ED8A  String in positive Binärzahl wandeln
ECC6 ED97 ED92  String in positive Binärzahl wandeln
FB21 FB58 FB58  String in Stringbereich forcieren
FB2E FB65 FB65  String in Stringbereich forcieren
FB8F FBB9 FBB9  String in Stringbereich kopieren
F88B            String kopieren
FB59 FB94 FB94  String kopieren, vom Stringstack löschen
F9E9 FA43 FA43  String und Byte holen
F828 F8D0 F8D0  String vom Stringstack löschen, ausgeben
E95C EA23 EA1E  String überlesen
F7CB F879 F879  String überlesen, auf Stringstack
CEA5 CF0C CF09  Stringausdruck holen
DAE7 DB06 DB02  Stringbearbeitungsroutine ausführen
F5E6            Stringbereich erweitern
F5CA F68F F68C  Stringbereich löschen
F5D1 F696 F693  Stringbereich-Platz reservieren
FBA6            Stringdescriptor kopieren
FBB3 FBCC FBCC  Stringdescriptorstack initialisieren
FB1B FB4D FB4D  Strings in Stringbereich forcieren
F897            Stringvergleich
CF1E CF88 CF85  Stringverknüpfung "+"
F863 F91D F91D  Stringverknüpfung "+"
FAD4            Suchstring in String suchen
C807 C8B5 C8B2  Synchronous Events bearbeiten
DA74 DA97 DA93  sämtliche Stringvariablen durchgehen
CC5B CD17 CD14  Tabelle der Fehlermeldungen
CF81 CFF0 CFED  Tabelle der Hierarchiecodes und Operatorenadressen
E64B E73B E736  Tabelle der Keywords ohne Buchstaben
DF30 E017 E012  Tabelle der Tokens mit Sonderteil
DFDC E0CB E0C6  Tabelle der Tokens mit Zeilennummer
CFF2            Tabelle für Operandenauswertung
```

```
F224 F2C8 F2C3  Tabelle für PRINT
EC2C            Tabelle für SAVE
CF07 CF70 CF6D  Teilausdruck holen
     F8DC F8DC  Teilstring ausgeben
F971 F9F3 F9F3  Teilstring holen
     DE26 DE21  Test auf "="
FF71 FF92 FF92  Test auf Buchstabe
FF7B FF9C FF9C  Test auf Buchstabe, Ziffer, "."
     DE1E DE19  Test auf Klammer auf
     DE22 DE1D  Test auf Klammer zu
     DE1A DE15  Test auf Komma
DD55 DE46 DE41  Test auf Komma
F501 F5B0 F5AB  Test auf Platz für Binärdatei
DD51 DE4C DE47  Test auf Statementende
FF3C FF5E FF5E  Test auf String, sonst Fehler
DD37 DE2A DE25  Test auf Zeichen nach Aufruf
FF83 FFA4 FFA4  Test auf Ziffer
FF7F FFA0 FFA0  Test auf Ziffer oder Dezimalpunkt
     F5E5 F5E0  Test, ob Adresse im Bereich liegt
     FC37 FC37  Test, ob Descriptor im Stringstack ist
     DBB6 DBB1  Text für INPUT holen und ausgeben
E313 E3DC E3D7  Token in Tabelle suchen
E2ED E3BD E3B8  Token suchen, Keywordadresse holen
DDEB DECF DECA  Trace-Routine
FF27 FF66 FF66  Typ des FAC holen, Flags setzen
FF45 FF66 FF66  Typ des FAC holen, Flags setzen
FF23 FF4B FF4B  Typ des FAC nach A holen
FE15 FE3B FE3B  Typen angleichen, Werte holen
EDCE EE9E EE99  unpacked BCD nach Binär wandeln
F706 F7E9 F7E9  User-Matrix neu setzen
E1E7 E2B8 E2B3  Variable auswerten
D686 D6C2 D6BF  Variable holen, ggf. neu anlegen
D690 D6CC D6C9  Variable holen, nicht anlegen
D6DE D71A D717  Variable suchen
E968 EA2F EA2A  Variable überlesen
D6D6 D712 D70F  Variable überlesen, Typ holen
F571            Variablen aus Stringbereich zurückholen
F549            Variablen in Stringbereich retten
C18C C178 C178  Variablen löschen
D76D D7A2 D79E  Variablen-Offset ins Programm speichern
D7DB D80A D806  Variablenadresse holen, auf Feld prüfen
D0FA D151 D14E  Variablenadresse nach FAC ("@")
D5AE D5ED D5EA  Variablenbereich freigeben
     F62E F629  Variablenbereich schützen
     F63E F640  Variablenbereich wieder ungeschützt
D906 D935 D931  Variablenname und Offset holen
D939 D970 D96C  Variablennamen auf Basic-Stack
DF89 E075 E070  Variablennamen auswerten
D92B D962 D95E  Variablennamen vom Basic-Stack
D731 D769 D765  Variablennamen überlesen
E989 EA52 EA4D  Variablenoffsets löschen
D97F D9B3 D9AF  Variablentyp entsprechend Token setzen
DFEA E0D6 E0D1  Variablentyp feststellen
D00D D077 D074  Variablenwert holen
CF59 CFC8 CFC5  Vergleichsoperator auswerten
D5C6 D605 D602  verkettete Listen der Felder löschen
D5BE D5FD D5FA  verkettete Listen der Variablen löschen
D999 D9CD D9C9  VL der Felder neu generieren
```

DACE	DAED	DAE9	VL durchgehen, Routine ausführen
F025	F100	F0FB	Vorkommastellenzahl ohne Sonderzeichenstellen holen
ED44	EE14	EE0F	Vorzeichen im String bestimmen
F069	F154	F14F	Vorzeichen setzen
FDA3	FDC4	FDC4	Vorzeichen von FAC holen
F09B			Vorzeichenflags holen
FF4B	FF6C	FF6C	Wert nach FAC kopieren
C312	C341	C33E	Window-Nummer holen
	C28C	C289	Window-Nummer transparent setzen
	D92B	D927	Word vom Basic-Stack holen
EFE1	F0B9	F0B4	Zahl bei letzter Stelle um 1 erhöhen
F2AF	F36E	F369	Zahl der Spaces MOD Ausgabebreite
EE9F	EF6F	EF6A	Zahl formatiert nach ASCII wandeln
EF0A	EFD5	EFD0	Zahl mit Nachkommastellen
EE8F	EF5F	EF5A	Zahl nach ASCII, kein positives Vorzeichen setzen
EE9D	EF6D	EF68	Zahl nach ASCII, maximal 9 Ziffern
F114			Zahl nach Binärstring wandeln
	F1E4	F1DF	Zahl nach Hex-/Binär-String wandeln
F119			Zahl nach Hex-String wandeln
EFC8	F09F	F09A	Zahl runden
FDAF	FDD5	FDD5	Zahl runden, nach FAC
C3B5	C3FF	C3FC	Zeichen an Drucker ausgeben
C3F8	C43B	C438	Zeichen an Kassette ausgeben
F06F			Zeichen ans Bufferende schreiben
C356	C3A3	C3A0	Zeichen ausgeben
C35C	C3AB	C3A8	Zeichen ausgeben
C377	C3C4	C3C1	Zeichen ausgeben (ohne LF-Behandlung)
C424			Zeichen einlesen
E145	E222	E21D	Zeichen für LIST ausgeben
DC9D	DC93	DC8E	Zeichen holen, " ", TAB, LF überlesen
DCA8	DC9E	DC99	Zeichen holen, CR/LF auswerten
E1FE	E2CF	E2CA	Zeichen in LIST-Buffer
DF25	E00D	E008	Zeichen in Token-Buffer
	C45F	C45C	Zeichen von Kassette lesen
C439	C472	C46F	Zeichen von Tastatur holen
C414			Zeichen zurück in Kassettenbuffer
FF1D	FF45	FF45	Zeiger auf FAC und Typ holen
		F714	Zeiger auf freien Basic-Bereich holen
E2DD	E3AD	E3A8	Zeiger in Keyword-Tabelle holen
EB48	EC1E	EC19	Zeile aus altem Programm kopieren
DF35	E01C	E017	Zeile bis Statementende übernehmen
DB1A	DB36	DB31	Zeile für LINE INPUT holen
E6D2	E7AA	E7A5	Zeile im Programm einfügen
E7A3	E869	E864	Zeile im Programm suchen
E79A	E861	E85C	Zeile suchen, ggf. Fehler ausgeben
DEBB	DFA9	DFA4	Zeile tokenisieren
CA4C	CB0D	CB0A	Zeile von Kassette holen
	E23D	E238	Zeile/Zeilennummer für AUTO nach ASCII wandeln
E767	E82C	E827	Zeilenadresse holen
C861	C914	C911	Zeilenadresse in Event-Block speichern
DDD2	DEB6	DEB1	Zeilenadresse nach HL holen
E687	E775	E770	Zeilenadressen im Programm eliminieren
E69D	E78B	E786	Zeilenadressen im Statement durch Zeilennummern ersetzen
CEE1	CF4B	CF48	Zeilennummer holen
E864	E925	E920	Zeilennummer im Statement ersetzen
DDD6	DEBA	DEB5	Zeilennummer/Direkt-Modus-Flag holen
EE04	EED4	EECF	Zeilennummern-String wandeln
CEB0	CF12	CF0F	Zeilennummernbereich holen

```
ED53 EE23 EE1E  Ziffernstring nach unpacked BCD wandeln
EE61 EF31 EF2C  Ziffernwert berechnen
F1F2 F29E F299  ZONE-Default setzen
E89F E960 E95B  zugehöriges ELSE suchen
C9C5 CA79 CA76  zugehöriges NEXT suchen
CA18 CACC CAC9  zugehöriges WEND suchen
     F959 F959  zwei Strings vom Stringstack
EB84 EC50 EC4B  Zwei-Byte-Wert von Kassette/Diskette laden
FF05 FF2D FF2D  Zweierkomplements-Byte in A nach Integer in FAC
```

5.5 RAM-Vektoren

5.5.1 Die Jump Restore-Vektoren, Haupttabelle

RAM	464	664	6128	Routine
BB00	19E0	1B5C	1B5C	KM INITIALIZE
BB03	1A1E	1B98	1B98	KM RESET
BB06	1A3C	1BBF	1BBF	KM WAIT CHAR
BB09	1A42	1BC5	1BC5	KM READ CHAR
BB0C	1A77	1BFA	1BFA	KM CHAR RETURN
BB0F	1ABD	1C46	1C46	KM SET EXPAND
BB12	1B2E	1CB3	1CB3	KM GET EXPAND
BB15	1A7B	1C04	1C04	KM EXP BUFFER RESET
BB18	1B56	1CDB	1CDB	KM WAIT KEY
BB1B	1B5C	1CE1	1CE1	KM READ KEY
BB1E	1CBD	1E45	1E45	KM TEST KEY
BB21	1BB3	1D38	1D38	KM GET STATE
BB24	1C5C	1DE5	1DE5	KM GET JOYSTICK
BB27	1D52	1ED8	1ED8	KM SET TRANSLATE
BB2A	1D3E	1EC4	1EC4	KM GET TRANSLATE
BB2D	1D57	1EDD	1EDD	KM SET SHIFT
BB30	1D43	1EC9	1EC9	KM GET SHIFT
BB33	1D5C	1EE2	1EE2	KM SET CTRL
BB36	1D48	1ECE	1ECE	KM GET CTRL
BB39	1CAB	1E34	1E34	KM SET REPEAT
BB3C	1CA6	1E2F	1E2F	KM GET REPEAT
BB3F	1C6D	1DF6	1DF6	KM SET DELAY
BB42	1C69	1DF2	1DF2	KM GET DELAY
BB45	1C71	1DFA	1DFA	KM ARM BREAK
BB48	1C82	1E0B	1E0B	KM DISARM BREAK
BB4B	1C90	1E19	1E19	KM BREAK EVENT
BB4E	1078	1070	1074	TXT INITIALIZE
BB51	1088	1080	1084	TXT RESET
BB54	1451	1455	1459	TXT VDU ENABLE
BB57	144B	144E	1452	TXT VDU DISABLE
BB5A	1400	13FA	13FE	TXT OUTPUT
BB5D	1334	1331	1335	TXT WR CHAR
BB60	13AB	13A8	13AC	TXT RD CHAR
BB63	13A7	13A4	13A8	TXT SET GRAPHIC
BB66	120C	1204	1208	TXT WIN ENABLE
BB69	1256	124E	1252	TXT GET WINDOW
BB6C	1540	154B	154F	TXT CLEAR WINDOW
BB6F	115E	1156	115A	TXT SET COLUMN
BB72	1169	1161	1165	TXT SET ROW
BB75	1174	116C	1170	TXT SET CURSOR
BB78	1180	1178	117C	TXT GET CURSOR
BB7B	1289	1282	1286	TXT CUR ENABLE
BB7E	129A	1293	1297	TXT CUR DISABLE
BB81	1279	1272	1276	TXT CUR ON
BB84	1281	127A	127E	TXT CUR OFF
BB87	11CE	11C6	11CA	TXT VALIDATE
BB8A	1268	1261	1265	TXT PLACE CURSOR
BB8D	1268	1261	1265	TXT REMOVE CURSOR

```
BB90 12A9 12A2 12A6  TXT SET PEN
BB93 12BD 12B6 12BA  TXT GET PEN
BB96 12AE 12A7 12AB  TXT SET PAPER
BB99 12C3 12BC 12C0  TXT GET PAPER
BB9C 12C9 12C2 12C6  TXT INVERSE
BB9F 137A 1377 137B  TXT SET BACK
BBA2 1387 1384 1388  TXT GET BACK
BBA5 12D3 12D0 12D4  TXT GET MATRIX
BBA8 12F1 12FE 1302  TXT SET MATRIX
BBAB 12FD 12FA 12FE  TXT SET M TABLE
BBAE 132A 1327 132B  TXT GET M TABLE
BBB1 14CB 14D0 14D4  TXT GET CONTROLS
BBB4 10E8 10E0 10E4  TXT STR SELECT
BBB7 1107 10FF 1103  TXT SWAP STREAMS
BBBA 15B0 15A4 15A8  GRA INITIALIZE
BBBD 15DF 15D3 15D7  GRA RESET
BBC0 15F4 15FA 15FE  GRA MOVE ABSOLUTE
BBC3 15F1 15F7 15FB  GRA MOVE RELATIVE
BBC6 15FC 1602 1606  GRA ASK CURSOR
BBC9 1604 160A 160E  GRA SET ORIGIN
BBCC 1612 1618 161C  GRA GET ORIGIN
BBCF 1734 16A1 16A5  GRA WIN WIDTH
BBD2 1779 16E6 16EA  GRA WIN HEIGHT
BBD5 17A6 1713 1717  GRA GET WINDOW WIDTH
BBD8 17BC 1729 172D  GRA GET WINDOW HEIGHT
BBDB 17C5 1732 1736  GRA CLEAR WINDOW
BBDE 17F6 1763 1767  GRA SET PEN
BBE1 1804 1771 1775  GRA GET PEN
BBE4 17FD 176A 176E  GRA SET PAPER
BBE7 180A 1776 177A  GRA GET PAPER
BBEA 1813 177F 1783  GRA PLOT ABSOLUTE
BBED 1810 177C 1780  GRA PLOT RELATIVE
BBF0 1827 1793 1797  GRA TEST ABSOLUTE
BBF3 1824 1790 1794  GRA TEST RELATIVE
BBF6 1839 17A5 17A9  GRA LINE ABSOLUTE
BBF9 1836 17A2 17A6  GRA LINE RELATIVE
BBFC 1945 193C 1940  GRA WR CHAR
BBFF 0AA0 0ABB 0ABF  SCR INITIALIZE
BC02 0AB1 0ACC 0AD0  SCR RESET
BC05 0B3C 0B33 0B37  SCR SET OFFSET
BC08 0B45 0B38 0B3C  SCR SET BASE
BC0B 0B50 0B52 0B56  SCR GET LOCATION
BC0E 0ACA 0AE5 0AE9  SCR SET MODE
BC11 0AEC 0B08 0B0C  SCR GET MODE
BC14 0AF7 0B13 0B17  SCR MODE CLEAR
BC17 0B57 0B59 0B5D  SCR CHAR LIMITS
BC1A 0B64 0B66 0B6A  SCR CHAR POSITION
BC1D 0BA9 0BAB 0BAF  SCR DOT POSITION
BC20 0BF9 0C01 0C05  SCR NEXT BYTE
BC23 0C05 0C0D 0C11  SCR PREV BYTE
BC26 0C13 0C1B 0C1F  SCR NEXT LINE
BC29 0C2D 0C35 0C39  SCR PREV LINE
BC2C 0C86 0C8A 0C8E  SCR INK ENCODE
BC2F 0CA0 0CA3 0CA7  SCR INK DECODE
BC32 0CEC 0CEE 0CF2  SCR SET INK
BC35 0D14 0D16 0D1A  SCR GET INK
BC38 0CF1 0CF3 0CF7  SCR SET BORDER
BC3B 0D19 0D1B 0D1F  SCR GET BORDER
```

```
BC3E 0CE4 0CE6 0CEA  SCR SET FLASHING
BC41 0CE8 0CEA 0CEE  SCR GET FLASHING
BC44 0DB3 0DB5 0DB9  SCR FILL BOX
BC47 0DB7 0DB9 0DBD  SCR FLOOD BOX
BC4A 0DDF 0DE1 0DE5  SCR CHAR INVERT
BC4D 0DFA 0DFC 0E00  SCR HARDWARE ROLL
BC50 0E3E 0E40 0E44  SCR SOFTWARE ROLL
BC53 0EF3 0EF5 0EF9  SCR UNPACK
BC56 0F49 0F26 0F2A  SCR REPACK
BC59 0C49 0C51 0C55  SCR ACCESS
BC5C 0C6B 0C70 0C74  SCR PIXELS
BC5F 0FC4 0F8F 0F93  SCR HORIZONTAL
BC62 102F 0F97 0F9B  SCR VERTICAL
BC65 2370 24BC 24BC  CAS INITIALIZE
BC68 237F 24CE 24CE  CAS SET SPEED
BC6B 238E 24E1 24E1  CAS NOISY
BC6E 2A4B 2BBB 2BBB  CAS START MOTOR
BC71 2A4F 2BBF 2BBF  CAS STOP MOTOR
BC74 2A51 2BC1 2BC1  CAS RESTORE MOTOR
BC77 2392 24E5 24E5  CAS IN OPEN
BC7A 23FC 2550 2550  CAS IN CLOSE
BC7D 2401 2557 2557  CAS IN ABANDON
BC80 2435 25A0 25A0  CAS IN CHAR
BC83 24AB 2618 2618  CAS IN DIRECT
BC86 249A 2607 2607  CAS RETURN
BC89 2496 2603 2603  CAS TEST EOF
BC8C 23AB 24FE 24FE  CAS OUT OPEN
BC8F 2415 257F 257F  CAS OUT CLOSE
BC92 242E 2599 2599  CAS OUT ABANDON
BC95 245B 25C6 25C6  CAS OUT CHAR
BC98 24EA 2653 2653  CAS OUT DIRECT
BC9B 2528 2692 2692  CAS CATALOG
BC9E 283F 29AF 29AF  CAS WRITE
BCA1 2836 29A6 29A6  CAS READ
BCA4 2851 29C1 29C1  CAS CHECK
BCA7 1E68 1FE9 1FE9  SOUND RESET
BCAA 1F9F 2114 2114  SOUND QUEUE
BCAD 206C 21CE 21CE  SOUND CHECK
BCB0 2089 21EB 21EB  SOUND ARM EVENT
BCB3 204A 21AC 21AC  SOUND RELEASE
BCB6 1ECB 2050 2050  SOUND HOLD
BCB9 1EE6 206B 206B  SOUND CONTINUE
BCBC 2338 2495 2495  SOUND AMPL ENVELOPE
BCBF 233D 249A 249A  SOUND TONE ENVELOPE
BCC2 2349 24A6 24A6  SOUND A ADDRESS
BCC5 234E 24AB 24AB  SOUND T ADDRESS
BCC8 005C 005C 005C  KL CHOKE OFF
BCCB 0329 0326 0326  KL ROM WALK
BCCE 0332 0330 0330  KL INIT BACK
BCD1 02A1 02A0 02A0  KL LOG EXT
BCD4 02B2 02B1 02B1  KL FIND COMMAND
BCD7 0163 0163 0163  KL NEW FRAME FLY
BCDA 016A 016A 016A  KL ADD FRAME FLY
BCDD 0170 0170 0170  KL DELETE FRAME FLY
BCE0 0176 0176 0176  KL NEW FAST TICKER
BCE3 017D 017D 017D  KL ADD FAST TICKER
BCE6 0183 0183 0183  KL DELETE FAST TICKER
BCE9 01B3 01B3 01B3  KL ADD TICKER
```

BCEC	01C5	01C5	01C5	KL DELETE TICKER
BCEF	01D2	01D2	01D2	KL INIT EVENT
BCF2	01E2	01E2	01E2	KL EVENT
BCF5	0228	0227	0227	KL SYNC RESET
BCF8	0285	0284	0284	KL DEL SYNCHRONOUS
BCFB	0256	0255	0255	KL NEXT SYNC
BCFE	021A	0219	0219	KL DO SYNC
BD01	0277	0276	0276	KL DONE SYNC
BD04	0295	0294	0294	KL EVENT DISABLE
BD07	029B	029A	029A	KL EVENT ENABLE
BD0A	028E	028D	028D	KL DISARM EVENT
BD0D	0099	0099	0099	KL TIME PLEASE
BD10	00A3	00A3	00A3	KL TIME SET
BD13	05DC	05D7	05ED	MC BOOT PROGRAM
BD16	060B	0609	061F	MC START PROGRAM
BD19	07BA	07A4	07B4	MC WAIT FLYBACK
BD1C	0776	0766	0776	MC SET MODE
BD1F	07C6	07B0	07C0	MC SCREEN OFFSET
BD22	0786	0776	0786	MC CLEAR INKS
BD25	0799	077C	078C	MC SET INKS
BD28	07E6	07D0	07E0	MC RESET PRINTER
BD2B	07F2	080B	081B	MC PRINT CHAR
BD2E	081B	0848	0858	MC BUSY PRINTER
BD31	0807	0834	0844	MC SEND PRINTER
BD34	0826	0853	0863	MC SOUND REGISTER
BD37	0888	08BB	08BD	JUMP RESTORE
BD3A		1D3C	1D3C	KM SET LOCKS
BD3D		1BFE	1BFE	KM FLUSH
BD40		145C	1460	TXT ASK STATE
BD43		15E8	15EC	GRA DEFAULT
BD46		19D1	19D5	GRA SET BACK
BD49		17AC	17B0	GRA SET FIRST
BD4C		17A8	17AC	GRA SET LINE MASK
BD4F		1626	162A	GRA FROM USER
BD52		19D5	19D9	GRA FILL
BD55		0B41	0B45	SCR SET POSITION
BD58		07FC	080C	MC PRINT TRANSLATION
BD5B			0397	KL RAM SELECT

5.5.2 Nebentabelle nach 464-Vektoren sortiert

464 RAM	464 ROM	664 RAM	664 ROM	6128 RAM	6128 ROM	
BD3A	2A98	BD5B	2C02	BD5E	2C02	EDIT
BD3D	2E18	BD5E	2F91	BD61	2F91	FLO Zahl kopieren
BD40	2E29	BD61	2F9F	BD64	2F9F	FLO INT nach REAL
BD43	2E55	BD64	2FC8	BD67	2FC8	FLO 4-Bytes nach REAL
BD46	2E66	BD67	2FD9	BD6A	2FD9	FLO REAL nach INTEGER
BD49	2E8E	BD6A	3001	BD6D	3001	FLO Zahl runden
BD4C	2EA1	BD6D	3014	BD70	3014	FLO FIX-Funktion
BD4F	2EAC	BD70	3055	BD73	3055	FLO INT-Funktion
BD52	2EB6	BD73	305F	BD76	305F	FLO Zahl für Dezimal-Wandlung aufbereiten
BD55	2F1D	BD76	30C6	BD79	30C6	FLO Zahl mit 10^A multiplizieren
BD58	333F	BD79	34A2	BD7C	34A2	FLO (HL):=(HL)+(DE)
BD5B	3337		349A		349A	FLO (HL):=(HL)-(DE)

BD5E	333B	BD7F	349E	BD82	349E	FLO (HL):=(DE)-(HL)
BD61	3415	BD82	3577	BD85	3577	FLO (HL):=(HL)*(DE)
BD64	349E	BD85	3604	BD88	3604	FLO (HL):=(HL)/(DE)
BD67	3578					FLO (HL):=(HL)*2^A
BD6A	359A	BD8B	36DF	BD8E	36DF	FLO Vergleich (HL)-(DE)
BD6D	35F8	BD8E	3731	BD91	3731	Vorzeichen invertieren
BD70	35E8	BD91	3727	BD94	3727	FLO SGN-Funktion
BD73	31AE	BD94	3345	BD97	3345	FLO DEG/RAD
BD76	31A3	BD97	2F73	BD9A	2F73	FLO PI-Funktion
BD79	310A	BD9A	32AC	BD9D	32AC	FLO SQR-Funktion
BD7C	310D	BD9D	32AF	BDA0	32AF	FLO Potenzierung
BD7F	3014	BDA0	31B6	BDA3	31B6	FLO LOG-Funktion
BD82	300F	BDA3	31B1	BDA6	31B1	FLO LOG10-Funktion
BD85	3090	BDA6	322F	BDA9	322F	FLO EXP-Funktion
BD88	31BC	BDA9	3353	BDAC	3353	FLO SIN-Funktion
BD8B	31B2	BDAC	3349	BDAF	3349	FLO COS-Funktion
BD8E	3231	BDAF	33C8	BDB2	33C8	FLO TAN-Funktion
BD91	3241	BDB2	33D8	BDB5	33D8	FLO ATN-Funktion
BD94	2E5E	BDB5	2FD1	BDB8	2FD1	FLO 5-Bytes nach REAL
BD97	2F94	BDB8	3136	BDBB	3136	FLO RND INITIALIZE
BD9A	2FA1	BDBB	3143	BDBE	3143	FLO RND SEED
BD9D	2FB7	BD7C	3159	BD7F	3159	FLO RND-Funktion
BDA0	2FE6	BD88	3188	BD8B	3188	FLO letzter RND-Wert
BDA3	3708		DD2F		DD2A	INT Zweierkomp-Zahl für Dezimal-Wandlung holen
BDA6	370E		DD35		DD30	INT Dezimal-Wandlung-Params für pos. Integer
BDA9	3715		DD3C		DD37	INT signed Binary nach Zweierkomplement
BDAC	3728		DD4F		DD4A	INT HL:=HL+DE
BDAF	3731		DD58		DD53	INT HL:=HL-DE
BDB2	3730		DD57		DD52	INT HL:=DE-HL
BDB5	3739		DD60		DD5B	INT HL:=HL*DE
BDB8	377A		DDA1		DD9C	INT HL:=HL DIV DE
BDBB	3781		DDA8		DDA3	INT HL:=HL MOD DE
BDBE	3750		DD77		DD72	vorzeichenlose Multiplikation
BDC1	378C		DDB3		DDAE	vorzeichenlose Division
BDC4	37E9		DE07		DE02	INT Vergleich HL-DE
BDC7	37D4		DDF2		DDED	INT HL:=-HL
BDCA	37E0		DDFE		DDF9	INT A:=SGN(HL)

5.5.3 Nebentabelle nach 664/6128-Vektoren sortiert

464 RAM	464 ROM	664 RAM	664 ROM	6128 RAM	6128 ROM	
BD3A	2A98	BD5B	2C02	BD5E	2C02	EDIT
BD3D	2E18	BD5E	2F91	BD61	2F91	FLO Zahl kopieren
BD40	2E29	BD61	2F9F	BD64	2F9F	FLO INT nach REAL
BD43	2E55	BD64	2FC8	BD67	2FC8	FLO 4-Bytes nach REAL
BD46	2E66	BD67	2FD9	BD6A	2FD9	FLO REAL nach INTEGER
BD49	2E8E	BD6A	3001	BD6D	3001	FLO Zahl runden
BD4C	2EA1	BD6D	3014	BD70	3014	FLO FIX-Funktion
BD4F	2EAC	BD70	3055	BD73	3055	FLO INT-Funktion
BD52	2EB6	BD73	305F	BD76	305F	FLO Zahl für Dezimal-Wandlung aufbereiten
BD55	2F1D	BD76	30C6	BD79	30C6	FLO Zahl mit 10^A multiplizieren
BD58	333F	BD79	34A2	BD7C	34A2	FLO (HL):=(HL)+(DE)
BD9D	2FB7	BD7C	3159	BD7F	3159	FLO RND-Funktion
BD5E	333B	BD7F	349E	BD82	349E	FLO (HL):=(DE)-(HL)

```
BD61 3415 BD82 3577 BD85 3577  FLO (HL):=(HL)*(DE)
BD64 349E BD85 3604 BD88 3604  FLO (HL):=(HL)/(DE)
BDA0 2FE6 BD88 3188 BD8B 3188  FLO letzter RND-Wert
BD6A 359A BD8B 36DF BD8E 36DF  FLO Vergleich (HL)-(DE)
BD6D 35F8 BD8E 3731 BD91 3731  Vorzeichen invertieren
BD70 35E8 BD91 3727 BD94 3727  FLO SGN-Funktion
BD73 31AE BD94 3345 BD97 3345  FLO DEG/RAD
BD76 31A3 BD97 2F73 BD9A 2F73  FLO PI-Funktion
BD79 310A BD9A 32AC BD9D 32AC  FLO SQR-Funktion
BD7C 310D BD9D 32AF BDA0 32AF  FLO Potenzierung
BD7F 3014 BDA0 31B6 BDA3 31B6  FLO LOG-Funktion
BD82 300F BDA3 31B1 BDA6 31B1  FLO LOG10-Funktion
BD85 3090 BDA6 322F BDA9 322F  FLO EXP-Funktion
BD88 31BC BDA9 3353 BDAC 3353  FLO SIN-Funktion
BD8B 31B2 BDAC 3349 BDAF 3349  FLO COS-Funktion
BD8E 3231 BDAF 33C8 BDB2 33C8  FLO TAN-Funktion
BD91 3241 BDB2 33D8 BDB5 33D8  FLO ATN-Funktion
BD94 2E5E BDB5 2FD1 BDB8 2FD1  FLO 5-Bytes nach REAL
BD97 2F94 BDB8 3136 BDBB 3136  FLO RND INITIALIZE
BD9A 2FA1 BDBB 3143 BDBE 3143  FLO RND SEED
```

5.5.4 Die Indirections

```
RAM  464  664  6128

BDCD 1263 125B 125F  TXT DRAW CURSOR
BDD0 1263 125B 125F  TXT UNDRAW CURSOR
BDD3 134A 1347 134B  TXT WRITE CHAR
BDD6 13C0 13BA 13BE  TXT UNWRITE
BDD9 140C 1406 140A  TXT OUT ACTION
BDDC 1816 1782 1786  GRA PLOT
BDDF 182A 1796 179A  GRA TEST
BDE2 183C 17B0 17B4  GRA LINE
BDE5 0C82 0C8C 0C90  SCR READ
BDE8 0C68 0C6D 0C71  SCR WRITE
BDEB 0AF7 0B13 0B17  SCR MODE CLEAR
BDEE 1C2F 1DB8 1DB8  KM TEST BREAK
BDF1 07F8 0825 0835  MC WAIT PRINTER
BDF4      1D40 1D40  KM SCAN KEYS
```

5.6 Die Basic-Tokens

Token	Bedeutung
00	Zeilenende
01	:
02	markierte Integervariable
03	markierte Stringvariable
04	markierte REAL-Variable
07	(??)
08	(??)
0B	unmarkierte Integervariable
0C	unmarkierte Stringvariable
0D	unmarkierte REAL-Variable
0D	unmarkierte Variable ohne festgestellten Typ
0E	Konstante 0
0F	Konstante 1
10	Konstante 2
11	Konstante 3
12	Konstante 4
13	Konstante 5
14	Konstante 6
15	Konstante 7
16	Konstante 8
17	Konstante 9
18	(Konstante 10)
19	1-Byte-Konstante
1A	2-Byte-Konstante
1B	2-Byte-Konstante, binär
1C	2-Byte-Konstante, hex
1D	Zeilenadresse (Zeiger auf Null am Zeilenende)
1E	Zeilennummer
1F	5-Byte-REAL-Wert
80	AFTER
81	AUTO
82	BORDER
83	CALL
84	CAT
85	CHAIN
86	CLEAR
87	CLG
88	CLOSEIN
89	CLOSEOUT
8A	CLS
8B	CONT
8C	DATA
8D	DEF
8E	DEFINT
8F	DEFREAL
90	DEFSTR
91	DEG
92	DELETE
93	DIM
94	DRAW
95	DRAWR
96	EDIT
0197	ELSE
98	END

```
99    ENT
9A    ENV
9B    ERASE
9C    ERROR
9D    EVERY
9E    FOR
9F    GOSUB
A0    GOTO
A1    IF
A2    INK
A3    INPUT
A4    KEY
A5    LET
A6    LINE
A7    LIST
A8    LOAD
A9    LOCATE
AA    MEMORY
AB    MERGE
AC    MID$
AD    MODE
AE    MOVE
AF    MOVER
B0    NEXT
B1    NEW
B2    ON
B3    ON BREAK
B4    ON ERROR GOTO 0
B5    ON SQ
B6    OPENIN
B7    OPENOUT
B8    ORIGIN
B9    OUT
BA    PAPER
BB    PEN
BC    PLOT
BD    PLOTR
BE    POKE
BF    PRINT
01C0 '
C1    RAD
C2    RANDOMIZE
C3    READ
C4    RELEASE
C5    REM
C6    RENUM
C7    RESTORE
C8    RESUME
C9    RETURN
CA    RUN
CB    SAVE
CC    SOUND
CD    SPEED
CE    STOP
CF    SYMBOL
D0    TAG
D1    TAGOFF
D2    TROFF
```

```
D3   TRON
D4   WAIT
D5   WEND
D6   WHILE
D7   WIDTH
D8   WINDOW
D9   WRITE
DA   ZONE
DB   DI
DC   EI
DD   FILL (nur CPC 664/6128)
DE   GRAPHICS (nur CPC 664/6128)
DF   MASK (nur CPC 664/6128)
E0   FRAME (nur CPC 664/6128)
E1   CURSOR (nur CPC 664/6128)
E3   ERL
E4   FN
E5   SPC
E6   STEP
E7   SWAP
EA   TAB
EB   THEN
EC   TO
ED   USING
EE   >
EF   =
F0   >=
F0   = >
F1   <
F2   <>
F3   <=
F3   = <
F4   +
F5   -
F6   *
F7   /
F8   ^
F9   \
FA   AND
FB   MOD
FC   OR
FD.  XOR
FE   NOT
FF00 ABS
FF01 ASC
FF02 ATN
FF03 CHR$
FF04 CINT
FF05 COS
FF06 CREAL
FF07 EXP
FF08 FIX
FF09 FRE
FF0A INKEY
FF0B INP
FF0C INT
FF0D JOY
FF0E LEN
```

```
FF0F LOG
FF10 LOG10
FF11 LOWER$
FF12 PEEK
FF13 REMAIN
FF14 SGN
FF15 SIN
FF16 SPACE$
FF17 SQ
FF18 SQR
FF19 STR$
FF1A TAN
FF1B UNT
FF1C UPPER$
FF1D VAL
FF40 EOF
FF41 ERR
FF42 HIMEM
FF43 INKEY$
FF44 PI
FF45 RND
FF46 TIME
FF47 XPOS
FF48 YPOS
FF49 DERR (nur CPC 664/6128)
FF71 BIN$
FF72 DEC$
FF73 HEX$
FF74 INSTR
FF75 LEFT$
FF76 MAX
FF77 MIN
FF78 POS
FF79 RIGHT$
FF7A ROUND
FF7B STRING$
FF7C TEST
FF7D TESTR
FF7E COPYCHR$ (nur CPC 664/6128)
FF7F VPOS
```

6 Die Listings der CPC-ROMs

In diesem Teil finden Sie die Listings der ROMs des CPC 464, des CPC 664 und des CPC 6128, jeweils getrennt nach Operating System und Basic. Nicht behandelt werden die DOS-ROMs von CPC 664 und CPC 6128. Die ROMs sind vollständig disassembliert und dokumentiert. Dort, wo sich kein Programm, sondern Tabellen oder andere Daten befanden, haben wir statt des Disassembler-Listings eine jeweils angemessene Repräsentation der Daten aufgenommen. Für die Dokumentation gilt, daß sie einen - hoffentlich gelungenen - Kompromiß zwischen der nötigen Ausführlichkeit und dem doch sehr eingeschränkten Platz darstellt, der zur Darstellung teilweise komplexer Zusammenhänge zur Verfügung stand. Um dennoch möglichst viel Informationen liefern zu können, mußten wir an vieien Stellen Gebrauch von Abkürzungen machen, was zweifellos die Lesbarkeit des Listings ein wenig herabsetzt. Wir hoffen aber, einen guten Mittelweg gefunden zu haben.

Wir haben versucht, die Darstellung der ROMs möglichst zu vereinheitlichen, um so die Konsistenz des Listings zu erhöhen. Für alle Listings haben wir daher folgende Standards eingeführt:

1. Wir haben die Listings in der allgemein, von Z80-Assemblern verwendeten Notation dargestellt. Darüberhinaus sind noch Sternchenreihen eingefügt, um den Anfang einer Routine zu markieren. Dies wurde bei allen eigenständigeren Routinen gemacht. Routinenteile, die nur einen Sinn innerhalb einer größeren Routine haben (so z.B. Austrittspunkte aus Routinen), haben keine eigene Sternchenreihe erhalten, sind aber durch eine Leerzeile von der Hauptroutine abgesetzt. Alle Routinen, die mit einer Sternchenreihe gekennzeichnet wurden, sind auch in den entsprechenden Routinentabellen aufgeführt. Auch andere Einträge, so z.B ASCII-Tabellen, sind mit Sternchenreihen versehen.

2. Bei Routinen, denen vom aufrufenden Programm Parameter übergeben werden müssen bzw. die an das aufrufende Programm Parameter zurück übergeben, haben wir nach der Routinenüberschrift (also nach der Sternchenreihe) einen IN/OUT-Block eingefügt. Dabei sind unter IN die Parameter gelistet, die der Routine übergeben werden müssen, unter OUT diejenigen, die sie zurück übergibt. Die Parameter werden nach dem Register bzw. der Speicherstelle, über die sie übergeben werden, und nach ihrer Bedeutung aufgeführt.

3. Vielfach ist es nicht sinnvoll, bei einer Reihe von Prozessorbefehlen jeden einzelnen Befehl zu dokumentieren, z.B. wenn zwei Words über den Akku addiert werden oder eine FLO-Mantisse verschoben wird. In solchen Fällen wurden die Befehle mit einer gemeinsamen Dokumentation versehen, was dadurch kenntlich gemacht ist, daß diese Texte um zwei Zeichen eingerückt sind.

6.1 Die Listings des CPC 464-ROMs

6.1.1 Das CPC-464-Betriebssystem

Das Betriebssystem des CPC ist in verschiedene semantische Abschnitte gegliedert, sogenannte Packs. Diese Packs beginnen im CPC 464 bei den folgenden Adressen:

1. Kernel (KL) $0000
2. Machine Pack (MC) $0580
3. Jump Restore $0888
4. Screen Pack (SCR) $0AA0
5. Text Screen Pack (TXT) $1078
6. Graphics Screen Pack (GRA) $15B0
7. Keyboard Manager (KM) $19E0
8. Sound Manager (SOUND) $1E68
9. Cassette Manager (CAS) $2370
10. Editor (EDIT) $2A98
11. Floating Point Arithmetics (FLO) $2E18
12. Integer Arithmetics (INT) $3708 (Zeichensatz $3800)

Im Listing sind die Packgrenzen jeweils durch eine Strichreihe gekennzeichnet.

Beim Listen des Betriebssystems mußten einige Besonderheiten berücksichtigt werden. So die Tatsache, daß ein Teil des Kernels (von $0391 bis $0579) in das RAM ab $B900 kopiert wird, um dort unabhängig vom Status der einzelnen ROMs immer verfügbar zu sein. Im Listing wurde dieses Segment mit der RAM-Adresse UND der ROM-Adresse aufgeführt.

Auch war die Routinentabelle des Jump Restore zu berücksichtigen. Sie wird ebenfalls in das RAM (ab $BB00) kopiert. Die einzelnen Spalten des Listings haben folgende Bedeutung: Die erste Spalte gibt die RAM-Adresse des jeweiligen Vektors an, die nächsten drei Spalten sind ein Hex-Dump des RAM. In der fünften Spalte steht die ROM-Adresse, die nächsten beiden Spalten sind ein Hex-Dump des ROM. Dann kommt der disassemblierte RAM-Vektor (mit jeweiliger Routinenadresse). In der letzten Spalte schließlich steht der Name der angesprungenen Routine.

```
--------------------------- KERNEL (KL) --------------------------

***********************************     RST0: System Reset
0000  01 89 7F     LD     BC,7F89           U ROM ausschalten,
0003  ED 49        OUT    (C),C             L ROM einschalten
0005  C3 80 05     JP     0580            System initialisieren

***********************************     RST1: LO JUMP
                                        (Sprung $0000-$3FFF)
0008  C3 82 B9     JP     B982

***********************************     LO PCHL
                                        (Sprung $0000-$3FFF, PC in HL)
000B  C3 7C B9     JP     B97C

***********************************     JP (BC)
000E  C5           PUSH   BC
000F  C9           RET

***********************************     RST2: LO SIDE CALL
                                        (Sprung $C000-$FFFF)
0010  C3 16 BA     JP     BA16

***********************************     KL SIDE PCHL
                                        (Sprung $C000-$FFFF, Adr. in HL)
0013  C3 10 BA     JP     BA10

***********************************     JP (DE)
0016  D5           PUSH   DE
0017  C9           RET

***********************************     RST3: LO FAR CALL
                                        (Sprung über gesamten Bereich in beliebige,
                                         auch externe, ROMs)
0018  C3 BF B9     JP     B9BF

***********************************     KL FAR PCHL
                                        (Sprung wie RST3, Adr. in HL)
001B  C3 B1 B9     JP     B9B1

***********************************     JP (HL)
001E  E9           JP     (HL)
001F  00           NOP

***********************************     RST4: RAM LAM
                                        (Byte aus RAM holen)
                                        IN : HL: Adresse
                                        OUT: A: Byte
0020  C3 CB BA     JP     BACB

***********************************     KL FAR CALL
                                        (wie LO FAR CALL, jedoch Adresse in HL)
                                        IN : HL: Adr. Aufrufadr. und Konfiguration
0023  C3 B9 B9     JP     B9B9
0026  00           NOP
0027  00           NOP
```

```
**********************************   RST5: FIRM JUMP
                                     (Sprung mit L ROM ein)
0028  C3 2E BA     JP     BA2E

002B  00                               Zwischenspeicher f. Konfig.
002C  ED 49        OUT    (C),C        Konfig. ausgeben
002E  D9           EXX                 1. Registersatz wiederanschalten
002F  FB           EI                  Interrupt erlauben

**********************************   RST6: USER
0030  F3           DI                  Interrupt ausschalten
0031  D9           EXX                 2. Registersatz an
0032  21 2B 00     LD     HL,002B      Zeiger auf Buffer f. Konfig.
0035  71           LD     (HL),C       Konfig. retten
0036  18 08        JR     0040

**********************************   RST7: INTERRUPT
0038  C3 39 B9     JP     B939

**********************************   EXT INTERRUPT VECTOR
003B  C9           RET

003C  00           NOP
003D  00           NOP
003E  00           NOP
003F  00           NOP

0040  CB D1        SET    2,C          L ROM ausschalten
0042  18 E8        JR     002C         und Schleife wieder durchlaufen

**********************************   Restore Hi Kernel Jumps
0044  21 40 00     LD     HL,0040        $0000-$003F
0047  2D           DEC    L              ins
0048  7E           LD     A,(HL)         RAM
0049  77           LD     (HL),A         kopieren
004A  20 FB        JR     NZ,0047
004C  3E C7        LD     A,C7         Opcode f. RST0
004E  32 30 00     LD     (0030),A     bei RST6 (USER) ablegen
0051  21 91 03     LD     HL,0391      von $0391
0054  11 00 B9     LD     DE,B900      nach $B900
0057  01 E9 01     LD     BC,01E9      $01E9 Bytes ins RAM
005A  ED B0        LDIR                kopieren

**********************************   KL CHOKE OFF
                                     OUT: C=00, wenn kein ext. ROM
                                          B: ROM No.
                                          DE: Einsprung in ROM
005C  F3           DI
005D  3A AB B1     LD     A,(B1AB)     lfd. Konfiguration
0060  ED 5B A9 B1  LD     DE,(B1A9)    Einsprung in lfd. Exp.ROM
0064  06 C0        LD     B,C0           $C0 Bytes
0066  21 00 B1     LD     HL,B100        ab $B100
0069  36 00        LD     (HL),00        löschen
006B  23           INC    HL
006C  10 FB        DJNZ   0069
006E  47           LD     B,A          Konfiguration
006F  0E FF        LD     C,FF         Flag
0071  A9           XOR    C            Konfig.=$FF (64K RAM)?
```

```
0072  C0            RET   NZ            sonst zurück
0073  4F            LD    C,A             durchgehend RAM? dann
0074  5F            LD    E,A             Flag und Einsprung
0075  57            LD    D,A             auf null setzen
0076  C9            RET

********************************** Reset cont'd 2
                                 IN : HL: Einsprung
                                         (=0 f. Basic)
                                       C: ROM-Konfiguration
                                 OUT: DE: Lo-RAM
                                       HL: Hi-RAM
                                       BC: Ende User-System-RAM
0077  7C            LD    A,H           Einsprung
0078  B5            OR    L             gleich null?
0079  79            LD    A,C           Konfiguration
007A  20 04         JR    NZ,0080       sonst weiter
007C  7D            LD    A,L           0 als Konfiguration
007D  21 06 C0      LD    HL,C006       als Default Einsprung
0080  32 A8 B1      LD    (B1A8),A      lfd. Konfiguration setzen
0083  32 AB B1      LD    (B1AB),A      Konfig. f. Sprung setzen
0086  22 A9 B1      LD    (B1A9),HL     Einsprungadresse setzen
0089  21 FF AB      LD    HL,ABFF       Hi-RAM
008C  11 40 00      LD    DE,0040       Lo-RAM
008F  01 FF B0      LD    BC,B0FF       Ende User-System-RAM
0092  31 00 C0      LD    SP,C000       Stackpointer Startwert
0095  DF            RST   18            Programm anspringen
0096  A9 B1                             bei (B1A9)
0098  C7            RST   00            bei Rückkehr Reset

********************************** KL TIME PLEASE
                                 OUT: DE,HL: 4 Bytes Timer
0099  F3            DI
009A  ED 5B 89 B1   LD    DE,(B189)     Timer Hi Word
009E  2A 87 B1      LD    HL,(B187)     Timer Lo Word
00A1  FB            EI
00A2  C9            RET

********************************** KL TIME SET
                                 IN : DE,HL: 4 Bytes Timer
00A3  F3            DI
00A4  AF            XOR   A             Sperrbyte
00A5  32 8B B1      LD    (B18B),A      setzen
00A8  ED 53 89 B1   LD    (B189),DE     Timer Hi setzen
00AC  22 87 B1      LD    (B187),HL     Timer Lo setzen
00AF  FB            EI
00B0  C9            RET

********************************** Scan Events
00B1  21 87 B1      LD    HL,B187       Timer
00B4  34            INC   (HL)          erhöhen
00B5  23            INC   HL            Zeiger auf nächstes Timer-Byte
00B6  28 FC         JR    Z,00B4        bei Übertrag dieses erhöhen
00B8  06 F5         LD    B,F5          Port B, PIO
00BA  ED 78         IN    A,(C)         laden
00BC  1F            RRA                 VSYNC ins Carry
00BD  30 08         JR    NC,00C7       kein VSYNC? dann weiter
00BF  2A 8C B1      LD    HL,(B18C)     Start Frame Fly Chain (FFC)
```

```
00C2  7C              LD     A,H            gibt es Einträge
00C3  B7              OR     A              in FFC ?
00C4  C4 53 01        CALL   NZ,0153      dann FFC bearbeiten
00C7  2A 8E B1        LD     HL,(B18E)    Start Fast Ticker Chain (FTC)
00CA  7C              LD     A,H            gibt es Einträge
00CB  B7              OR     A              in FTC?
00CC  C4 53 01        CALL   NZ,0153      dann FTC bearbeiten
00CF  CD 61 1F        CALL   1F61         Sound Queues durchgehen
00D2  21 92 B1        LD     HL,B192      Frequenzteiler f. Ticker Chain
00D5  35              DEC    (HL)         herunterzählen
00D6  C0              RET    NZ           <>0? dann zurück
00D7  36 06           LD     (HL),06      sonst TC-Zähler auf Startwert
00D9  CD B7 1B        CALL   1BB7         Tastatur abfragen
00DC  2A 90 B1        LD     HL,(B190)    Start Ticker Chain (TC)
00DF  7C              LD     A,H            gibt es Einträge
00E0  B7              OR     A              in der TC?
00E1  C8              RET    Z            sonst (TC leer) raus
00E2  21 04 B1        LD     HL,B104      sonst Interrupt-Flag (INTFLG)
00E5  CB C6           SET    0,(HL)       b0 f. TC bearbeiten setzen
00E7  C9              RET

************************************       async, not Expr. Events einhängen
                                           IN : HL: Zeiger auf PQ-Zähler
00E8  2B              DEC    HL           Zeiger auf KAPQ
00E9  36 00           LD     (HL),00      Hi-Byte löschen
00EB  2B              DEC    HL           Zeiger auf das Lo-Byte d. KAPQ
00EC  3A 01 B1        LD     A,(B101)     Start d. async. Pend. Queue
00EF  B7              OR     A            bereits Einträge in der APQ?
00F0  20 0C           JR     NZ,00FE      dann in bestehende Queue einh.
00F2  22 00 B1        LD     (B100),HL    sonst lfd. Event als APQ-Start
00F5  22 02 B1        LD     (B102),HL    und als Ende d. APQ setzen
00F8  21 04 B1        LD     HL,B104      Interrupt-Flag
00FB  CB F6           SET    6,(HL)       b6 setzen f. APQ nicht leer
00FD  C9              RET

00FE  ED 5B 02 B1     LD     DE,(B102)    letzter APQ-Eintrag
0102  22 02 B1        LD     (B102),HL    Adr. lfd. Event dafür setzen
0105  EB              EX     DE,HL
0106  73              LD     (HL),E         und Adr. lfd. Event
0107  23              INC    HL             in die KAPQ des alten
0108  72              LD     (HL),D         Events speichern
0109  C9              RET

************************************       async. PQ und Ticker Chain bearb.
010A  ED 73 05 B1     LD     (B105),SP      Stackpointer retten
010E  31 87 B1        LD     SP,B187
0111  E5              PUSH   HL             Register
0112  D5              PUSH   DE             retten
0113  C5              PUSH   BC
0114  21 04 B1        LD     HL,B104      Interrupt Flag
0117  CB 76           BIT    6,(HL)       Einträge in APQ?
0119  28 1E           JR     Z,0139       sonst TC bearbeiten
011B  CB FE           SET    7,(HL)       b7 INTFLG:=1 f. APQ in Bearb.
011D  2A 00 B1        LD     HL,(B100)    Start d. APQ
0120  7C              LD     A,H            APQ
0121  B7              OR     A              zuende?
0122  28 0E           JR     Z,0132       dann ggf. TC bearbeiten
0124  5E              LD     E,(HL)         KA d. lfd. Eintrags
```

```
0125  23          INC   HL            in der APQ
0126  56          LD    D,(HL)        nach DE
0127  ED 53 00 B1 LD    (B100),DE    und als neuen APQ-Start setzen
012B  23          INC   HL           Zeiger auf PQ-Zähler
012C  CD 0A 02    CALL  020A         Event ausführen
012F  F3          DI
0130  18 EB       JR    011D         nächsten APQ-Eintrag bearb.
0132  21 04 B1    LD    HL,B104      Interrupt-Flag
0135  CB 46       BIT   0,(HL)       Ticker aktiv?
0137  28 10       JR    Z,0149       sonst raus
0139  36 00       LD    (HL),00      Interrupt-Flags löschen
013B  37          SCF                CY:=1 f. 'Interrupt laufend'
013C  08          EX    AF,AF'       ins Interrupt Set
013D  CD 89 01    CALL  0189         Ticker Chain bearbeiten
0140  B7          OR    A            CY:=0 f. 'Interrupt frei'
0141  08          EX    AF,AF'       ins Interrupt Set
0142  21 04 B1    LD    HL,B104      Interrupt Flags
0145  7E          LD    A,(HL)       laden
0146  B7          OR    A            und testen
0147  20 D2       JR    NZ,011B      APQ-Einträge? d. ausführen
0149  36 00       LD    (HL),00      sonst Interrupt Flags löschen
014B  C1          POP   BC             Register
014C  D1          POP   DE             wiederherstellen
014D  E1          POP   HL
014E  ED 7B 05 B1 LD    SP,(B105)    Stackpointer wieder zurück
0152  C9          RET

*********************************  KL KICK EVENT
                                   IN : HL: Start d. Chain (FFC/FTC)
0153  5E          LD    E,(HL)         KA innerhalb
0154  23          INC   HL             der Chain
0155  7E          LD    A,(HL)         nach A,E laden
0156  23          INC   HL           Zeiger auf PQ-Zähler
0157  B7          OR    A            letzter Eintrag?
0158  CA E2 01    JP    Z,01E2       d. einhängen, raus
015B  57          LD    D,A          KA nach DE
015C  D5          PUSH  DE           und retten
015D  CD E2 01    CALL  01E2         lfd. Event einhängen
0160  E1          POP   HL           Zeiger nächster Event nach HL
0161  18 F0       JR    0153         und nächsten Event bearbeiten

*********************************  KL NEW FRAME FLY
                                   IN : HL: Zeiger auf KAFFC
                                          DE: Routinenadresse
                                          B: Priority Byte
                                          C: ROM-Konfiguration
0163  E5          PUSH  HL           Zeiger auf KAFFC
0164  23          INC   HL             Zeiger auf
0165  23          INC   HL             den PQ-Zähler
0166  CD D2 01    CALL  01D2         Event Block aufbauen
0169  E1          POP   HL           Zeiger auf KAFFC

*********************************  KL ADD FRAME FLY
                                   IN : HL: Zeiger auf KAFFC
016A  11 8C B1    LD    DE,B18C      Start der FFC
016D  C3 73 03    JP    0373         Event in FFC einhängen
```

```
*********************************** KL DELETE FRAME FLY
                                    IN : HL: Zeiger auf KAFFC
                                    OUT: CY=0, wenn Ev. nicht in FFC
0170  11 8C B1     LD     DE,B18C     Start der FFC
0173  C3 82 03     JP     0382        Event aus FFC aushängen


*********************************** KL NEW FAST TICKER
                                    IN : HL: Zeiger auf KAFTC
                                         DE: Routinenadresse
                                         B: Priority Byte
                                         C: ROM-Konfiguration
0176  E5           PUSH   HL          Zeiger auf KAFTC
0177  23           INC    HL            Zeiger auf
0178  23           INC    HL            den PQ-Zähler
0179  CD D2 01     CALL   01D2        Event-Block aufbauen
017C  E1           POP    HL          Zeiger auf KAFTC

*********************************** KL ADD FAST TICKER
                                    IN : HL: Zeiger auf KAFTC
017D  11 8E B1     LD     DE,B18E     Start der FTC
0180  C3 73 03     JP     0373        Event in FTC einhängen

*********************************** KL DELETE FAST TICKER
                                    IN : HL: Zeiger auf KAFTC
                                    OUT: CY=0, wenn Ev. nicht in FTC
0183  11 8E B1     LD     DE,B18E     Start der FTC
0186  C3 82 03     JP     0382        Event aus FTC aushängen

*********************************** Ticker Chain bearbeiten
0189  2A 90 B1     LD     HL,(B190)   1. Eintrag in Ticker Chain
018C  7C           LD     A,H           Ticker Chain
018D  B7           OR     A             zuende?
018E  C8           RET    Z           dann raus
018F  5E           LD     E,(HL)        Koppeladr. d.
0190  23           INC    HL            lfd. Eintrags
0191  56           LD     D,(HL)        in TC nach DE
0192  23           INC    HL
0193  4E           LD     C,(HL)        Tick Count
0194  23           INC    HL            nach BC
0195  46           LD     B,(HL)
0196  78           LD     A,B           Tick Count
0197  B1           OR     C             gleich null?
0198  28 16        JR     Z,01B0      d. nächsten Eintrag bearbeiten
019A  0B           DEC    BC          Tick Count erniedrigen
019B  78           LD     A,B           Tick Count
019C  B1           OR     C             ungleich null?
019D  20 0E        JR     NZ,01AD     d. speichern, nächst. Eintrag
019F  D5           PUSH   DE          KATC retten
01A0  23           INC    HL            HL: Zeiger auf
01A1  23           INC    HL            Hi-Byte d. Reload Counts
01A2  E5           PUSH   HL          retten
01A3  23           INC    HL          Zeiger auf KAPQ
01A4  CD E2 01     CALL   01E2        Event einhängen
01A7  E1           POP    HL          Zeiger auf Reload Count, Hi
01A8  46           LD     B,(HL)        Reload Count
01A9  2B           DEC    HL            nach BC laden
01AA  4E           LD     C,(HL)
```

```
01AB  2B         DEC   HL
01AC  D1         POP   DE          Zeiger auf nächsten Eintrag
01AD  70         LD    (HL),B      Reload Count
01AE  2B         DEC   HL          in Ticker Kopf
01AF  71         LD    (HL),C      speichern
01B0  EB         EX    DE,HL       Adr. d. nächsten Eintr. n. HL
01B1  18 D9      JR    018C        Ticker Chain weiter bearbeiten

********************************** KL ADD TICKER
                                 IN : HL: Zeiger auf KATC
                                      DE: Tick Count
                                      BC: Reload Count
01B3  E5         PUSH  HL          Zeiger auf KATC
01B4  23         INC   HL            Zeiger auf
01B5  23         INC   HL            den Tick Count
01B6  F3         DI
01B7  73         LD    (HL),E        Tick Count
01B8  23         INC   HL            setzen
01B9  72         LD    (HL),D
01BA  23         INC   HL
01BB  71         LD    (HL),C        Reload Count
01BC  23         INC   HL            setzen
01BD  70         LD    (HL),B
01BE  E1         POP   HL          Zeiger auf KATC.
01BF  11 90 B1   LD    DE,B190     Start Ticker Chain
01C2  C3 73 03   JP    0373        Event in Ticker Chain einhäng.

********************************** KL DELETE TICKER
                                 IN : HL: Zeiger auf KATC
                                 OUT: DE: lfd. TC-Zähler
                                      CY=0, wenn Event nicht in TC
01C5  11 90 B1   LD    DE,B190     Start Ticker Chain
01C8  CD 82 03   CALL  0382        Event aus TC aushängen
01CB  D0         RET   NC          war nicht drin? d. raus
01CC  EB         EX    DE,HL       Zeiger auf KATC, Hi nach HL
01CD  23         INC   HL          Zeiger auf TC-Zähler
01CE  5E         LD    E,(HL)
01CF  23         INC   HL            lfd. TC-Zähler nach DE
01D0  56         LD    D,(HL)
01D1  C9         RET

********************************** KL INIT EVENT
                                 IN : HL: Zeiger auf KAPQ
                                      DE: Routinenadresse
                                      B: Priority Byte
                                      C: ROM-Konfiguration
                                 OUT: HL: Zeiger auf User-Area
01D2  F3         DI
01D3  23         INC   HL            Zeiger auf
01D4  23         INC   HL            PQ-Count
01D5  36 00      LD    (HL),00     PQ-Count zurücksetzen
01D7  23         INC   HL          Priority Byte
01D8  70         LD    (HL),B      setzen
01D9  23         INC   HL            Routinenadresse
01DA  73         LD    (HL),E        setzen
01DB  23         INC   HL
01DC  72         LD    (HL),D
01DD  23         INC   HL          ROM-Konfiguration
```

```
01DE  71          LD    (HL),C       setzen
01DF  23          INC   HL           Zeiger auf User-Area
01E0  FB          EI
01E1  C9          RET

***********************************  KL EVENT
                                     IN : HL: Zeiger auf KAPQ
01E2  23          INC   HL              Zeiger
01E3  23          INC   HL              auf PQ-Zähler
01E4  F3          DI
01E5  7E          LD    A,(HL)       PQ-Zähler laden
01E6  34          INC   (HL)         und erhöhen
01E7  FA 06 02    JP    M,0206       war zw. $7F u. $FE? d. Count-1
01EA  B7          OR    A            war <>0? ($01..$7E, $FF)
01EB  20 13       JR    NZ,0200      d. Count erhöht lassen, raus
01ED  23          INC   HL           Zeiger auf Priority Byte
01EE  7E          LD    A,(HL)       laden
01EF  2B          DEC   HL           Zeiger auf PQ-Count
01F0  B7          OR    A            Event synchronous?
01F1  F2 2F 02    JP    P,022F       d. (n. Prior. geordn.) einhän.
01F4  08          EX    AF,AF'       nicht im Interrupt?
01F5  30 12       JR    NC,0209      dann async. Event ausführen
01F7  08          EX    AF,AF'
01F8  87          ADD   A            Express Event?
01F9  F2 E8 00    JP    P,00E8       sonst async. Event einhängen
01FC  23          INC   HL             Zeiger
01FD  23          INC   HL             auf Routinenadresse
01FE  18 23       JR    0223         und Routine ausführen
0200  08          EX    AF,AF'         Interrupt nur wieder
0201  38 01       JR    C,0204         zulassen,
0203  FB          EI                   wenn Interruptbehandlung
0204  08          EX    AF,AF'         nicht laufend
0205  C9          RET

0206  35          DEC   (HL)         Zähler wieder zurücksetzen
0207  18 F7       JR    0200         und raus

0209  08          EX    AF,AF'

***********************************  Event wiederholt ausführen
                                     IN : HL: Zeiger auf PQ-Zähler
020A  FB          EI
020B  7E          LD    A,(HL)         Zähler laden
020C  B7          OR    A              >127?
020D  F8          RET   M              dann raus, Event 'geschützt'
020E  E5          PUSH  HL             Zeiger auf Zähler retten
020F  CD 1C 02    CALL  021C           Event ausführen
0212  E1          POP   HL             Zeiger auf Zähler
0213  35          DEC   (HL)           Zähler dekrementieren
0214  C8          RET   Z              null? dann raus
0215  F2 0E 02    JP    P,020E         Count war <>0? d. weiter
0218  34          INC   (HL)           sonst Count wieder :=0 setzen
0219  C9          RET                  raus

***********************************  KL DO SYNC
                                     IN : HL: Zeiger KAPQ
021A  23          INC   HL              Zeiger auf
021B  23          INC   HL              PQ-Count berechnen
```

```
***********************************  Event ausführen
                                     IN : HL: Zeiger auf PQ-Count
021C  23          INC    HL             Zeiger auf Priority Byte
021D  7E          LD     A,(HL)         Priority Byte laden
021E  23          INC    HL             Zeiger auf Routinenadresse
021F  1F          RRA                   Far Call?
0220  D2 B9 B9    JP     NC,B9B9        dann entspr. behandeln

***********************************  Near Address Routine ausführen
                                     IN : HL: Zeiger auf Routinenadr.
0223  5E          LD     E,(HL)           Routinenadresse
0224  23          INC    HL               nach DE
0225  56          LD     D,(HL)
0226  EB          EX     DE,HL          Routinenadresse nach HL
0227  E9          JP     (HL)           Routine ausführen, danach raus

***********************************  KL SYNC RESET
0228  21 00 00    LD     HL,0000          lfd. Priorität und
022B  22 94 B1    LD     (B194),HL        SPQ/Sperr-Priorität löschen
022E  C9          RET

***********************************  Synchronous Event einhängen
                                     IN : HL: Zeiger auf PQ-Zähler
                                           A: Priority Byte
022F  E5          PUSH   HL             Zeiger auf Zähler retten
0230  47          LD     B,A            Priority Byte
0231  11 96 B1    LD     DE,B196        Start d. Sync. PQ (SPQ) +3
0234  EB          EX     DE,HL          Zeiger auf KAPQ +3 nach HL
0235  2B          DEC    HL               Zeiger auf
0236  2B          DEC    HL               Koppeladresse, Hi
0237  56          LD     D,(HL)           KAPQ nach DE
0238  2B          DEC    HL               laden
0239  5E          LD     E,(HL)
023A  7A          LD     A,D            Hi-Byte d. Koppeladresse
023B  B7          OR     A              letzter Eintrag in SPQ?
023C  28 07       JR     Z,0245         d. lfd. Event Block anhängen
023E  13          INC    DE               Zeiger auf folgendes
023F  13          INC    DE               (neues) Priority Byte
0240  13          INC    DE               berechnen
0241  1A          LD     A,(DE)         Priority Byte laden
0242  B8          CP     B              mit lfd. Priorität vergleichen
0243  30 EF       JR     NC,0234        alte >= neue Prior.? d. weiter
0245  D1          POP    DE             sonst Zeiger auf lfd. Event
0246  1B          DEC    DE             Zeiger auf KA, Hi
0247  23          INC    HL             Zeiger auf letzte KAPQ, Hi
0248  7E          LD     A,(HL)           letzte KAPQ in lfd.
0249  12          LD     (DE),A           Event Block eintragen,
024A  1B          DEC    DE               dabei Adr. d. lfd.
024B  72          LD     (HL),D           Event Blocks in letzte
024C  2B          DEC    HL               KAPQ eintragen,
024D  7E          LD     A,(HL)           lfd. Event also einhängen
024E  12          LD     (DE),A
024F  73          LD     (HL),E
0250  08          EX     AF,AF'           Interrupt nur zulassen
0251  38 01       JR     C,0254           wenn Interruptbehandlung
0253  FB          EI                      nicht laufend
0254  08          EX     AF,AF'
0255  C9          RET
```

```
************************************  KL NEXT SYNC
                                      OUT: A: alte Priorität
                                           HL: alte Event-Adresse
                                           CY:=1, wenn nächster Event
0256  F3           DI
0257  2A 93 B1     LD    HL,(B193)      Adr. d. lfd. Events
025A  7C           LD    A,H              kein Eintrag mehr
025B  B7           OR    A                in SPQ?
025C  28 17        JR    Z,0275         dann raus
025E  E5           PUSH  HL             Zeiger auf neuen Event Block
025F  5E           LD    E,(HL)           Koppeladresse laden,
0260  23           INC   HL               d.h. Zeiger auf folgenden
0261  56           LD    D,(HL)           Event nach DE
0262  23           INC   HL
0263  23           INC   HL             Zeiger auf Priority Byte
0264  3A 95 B1     LD    A,(B195)       Priorität d. lfd. Events laden
0267  BE           CP    (HL)           und m. neuer Priorität vergl.
0268  30 0A        JR    NC,0274        lfd. Prior. größer? d. raus
026A  F5           PUSH  AF             lfd. Priorität retten
026B  7E           LD    A,(HL)         neue Priorität laden
026C  32 95 B1     LD    (B195),A       und als lfd. setzen
026F  ED 53 93 B1  LD    (B193),DE      neuen Event als lfd. setzen
0273  F1           POP   AF             alte Priorität
0274  E1           POP   HL             Adr. d. alten Events
0275  FB           EI
0276  C9           RET

************************************  KL DONE SYNC
                                      IN : HL: Zeiger Event
                                           A: Priorität d. Events
0277  32 95 B1     LD    (B195),A       Priorität als lfd. setzen
027A  23           INC   HL               Zeiger auf
027B  23           INC   HL               PQ-Zähler
027C  35           DEC   (HL)           Zähler erniedrigen
027D  C8           RET   Z              schon null? dann raus
027E  F3           DI
027F  F2 2F 02     JP    P,022F         sonst b7=0? d. einhängen
0282  34           INC   (HL)           sonst Count wiederherstellen
0283  FB           EI
0284  C9           RET

************************************  KL DEL SYNCHRONOUS
                                      IN : HL: Zeiger a. Event, KAPQ
0285  CD 8E 02     CALL  028E           Event abschalten
0288  11 93 B1     LD    DE,B193        und aus Sync. Pending Queue
028B  C3 82 03     JP    0382           aushängen

************************************  KL DISARM EVENT
                                      IN : HL: Zeiger a. Event, KAPQ
                                      OUT: HL unverändert
028E  23           INC   HL               Zeiger auf
028F  23           INC   HL               PQ-Zähler
0290  36 C0        LD    (HL),C0        Zähler f. Event inaktiv setzen
0292  2B           DEC   HL               Zeiger auf Event
0293  2B           DEC   HL               wiederherstellen
0294  C9           RET
```

```
***********************************  KL EVENT DISABLE
0295  21 95 B1     LD     HL,B195      Priority Byte lfd. Event
0298  CB EE        SET    5,(HL)       auf max. Priorität setzen
029A  C9           RET

***********************************  KL EVENT ENABLE
029B  21 95 B1     LD     HL,B195      Priority Byte lfd. Event
029E  CB AE        RES    5,(HL)       wieder normale Priorität setz.
02A0  C9           RET

***********************************  KL LOG EXT
                                     IN : HL: Zieladr. f. Eintrag
                                           C: ROM-No.
02A1  E5           PUSH   HL           Zieladresse retten
02A2  ED 5B A6 B1  LD     DE,(B1A6)    Adr. d. 1. Eintrags in VL
02A6  22 A6 B1     LD     (B1A6),HL    neu auf lfd. Eintrag setzen
02A9  73           LD     (HL),E         Adr. d. alten Eintrags
02AA  23           INC    HL             als Koppeladr. setzen
02AB  72           LD     (HL),D
02AC  23           INC    HL
02AD  71           LD     (HL),C       ROM-Nummer setzen
02AE  23           INC    HL
02AF  70           LD     (HL),B
02B0  E1           POP    HL
02B1  C9           RET

***********************************  KL FIND COMMAND
                                     IN : HL: Zeiger auf Befehlsstring
                                     OUT: HL: Aufrufadresse
                                           C: ROM-Konfiguration
                                           CY:=1, wenn String gefunden
02B2  11 96 B1     LD     DE,B196      Zeiger auf RSX-String
02B5  01 10 00     LD     BC,0010      maximale Länge
02B8  CD A6 BA     CALL   BAA6         RSX-String nach $B196 kopieren
02BB  EB           EX     DE,HL        Zeiger hinter die Kopie -> HL
02BC  2B           DEC    HL           Zeiger auf letztes Zeichen
02BD  CB FE        SET    7,(HL)       ggf. als letztes Zeichen setz.
02BF  2A A6 B1     LD     HL,(B1A6)    Adr. d. 1. Eintrags in VL
02C2  7D           LD     A,L          Lo-Byte
02C3  18 10        JR     02D5         ans Schleifenende springen

02C5  E5           PUSH   HL           Koppeladr. retten
02C6  23           INC    HL             Zeiger auf
02C7  23           INC    HL             ROM-Nummer aus VL-Eintrag
02C8  4E           LD     C,(HL)       ROM-Nummer laden
02C9  23           INC    HL
02CA  46           LD     B,(HL)
02CB  CD F4 02     CALL   02F4         RSX-Str. in entspr. ROM suchen
02CE  D1           POP    DE           Zeiger auf Eintrag
02CF  D8           RET    C            String gefunden? d. raus
02D0  EB           EX     DE,HL        Zeiger auf Koppeladr.
02D1  7E           LD     A,(HL)         Koppeladresse
02D2  23           INC    HL             nach HL
02D3  66           LD     H,(HL)         laden
02D4  6F           LD     L,A
02D5  B4           OR     H            letzter Eintrag in VL?
02D6  20 ED        JR     NZ,02C5      sonst nächsten Eintrag bearb.
```

```
02D8  0E FF     LD    C,FF      nicht gefund.? d. Anfangsno.-1
02DA  0C        INC   C         ROM-Nummer erhöhen
02DB  CD 83 BA  CALL  BA83      ROM-Kennung holen
02DE  F5        PUSH  AF        und retten
02DF  E6 03     AND   03        b1,b0 isolieren
02E1  47        LD    B,A       und nach B retten
02E2  CC F4 02  CALL  Z,02F4    b1=b0=0? d. Str. in ROM suchen
02E5  38 09     JR    C,02F0    gefunden? d. System starten
02E7  F1        POP   AF        Kennung aus ROM
02E8  87        ADD   A         b7=1?
02E9  30 EF     JR    NC,02DA   sonst nächstes ROM bearbeiten
02EB  79        LD    A,C       ROM-No.
02EC  B7        OR    A         null?
02ED  28 EB     JR    Z,02DA    dann nächstes ROM bearbeiten
02EF  C9        RET             raus, wenn ROM-No.<>0, b7=1
02F0  F1        POP   AF        ROM-Kennung
02F1  C3 0B 06  JP    060B      System starten

***********************************  RSX-String in ROM suchen
                                     IN : BC: ROM-No. bzw. Tab.Adr.
                                     OUT: HL: Einsprung f. Kommando
                                          DE: Zeiger hinter Kommando
                                          CY:=1, wenn String gefunden
02F4  21 04 C0  LD    HL,C004   Zeiger auf Adr. d. RSX-Strings
02F7  78        LD    A,B       Hi-Byte d. Tabellenadresse
02F8  B7        OR    A         gleich null?
02F9  28 04     JR    Z,02FF    d. C ROM-No., $C004 a. Default
02FB  60        LD    H,B         sonst Zeiger auf RSX-Tabelle
02FC  69        LD    L,C         nach HL laden
02FD  0E FF     LD    C,FF      Default ROM-No.: Basic
02FF  CD 7E BA  CALL  BA7E      ROM auswählen u. anschalten
0302  C5        PUSH  BC        ROM-No. retten
0303  5E        LD    E,(HL)      Zeiger auf RSX-
0304  23        INC   HL          Stringtabelle
0305  56        LD    D,(HL)      nach DE
0306  23        INC   HL        Zeiger auf Adressentabelle
0307  EB        EX    DE,HL     vertauschen
0308  18 17     JR    0321      in Schleifenende springen

030A  01 96 B1  LD    BC,B196   Zeiger auf Befehlsstring
030D  0A        LD    A,(BC)    Zeichen aus Befehlsstring
030E  BE        CP    (HL)      mit Zeichen aus Tabelle vergl.
030F  20 08     JR    NZ,0319   ungleich? dann nächsten String
0311  23        INC   HL        sonst Zeiger nächstes Zeichen
0312  03        INC   BC        setzen
0313  87        ADD   A         letztes Zeichen des Befehls?
0314  30 F7     JR    NC,030D   sonst weiter vergleichen
0316  EB        EX    DE,HL     d. Einsprungadresse nach HL
0317  18 0C     JR    0325      und raus
0319  7E        LD    A,(HL)      bei Ungleichheit bis
031A  23        INC   HL          Stringende in ROM-Tabelle
031B  87        ADD   A           lesen
031C  30 FB     JR    NC,0319
031E  13        INC   DE          Zeiger auf nächsten
031F  13        INC   DE          RSX-Einsprung
0320  13        INC   DE          setzen
0321  7E        LD    A,(HL)    1. Zeichen d. nächsten Strings
0322  B7        OR    A         Tabellenende?
```

```
0323  20 E5       JR    NZ,030A      sonst weiter suchen
0325  C1          POP   BC           alte Konfiguration
0326  C3 8C BA    JP    BA8C         wieder setzen

*********************************  KL ROM WALK
                                   IN : DE: Lo RAM
                                        HL: Hi RAM
                                   OUT: DE: Lo RAM, neu
                                        HL: Hi RAM, neu
0329  0E 07       LD    C,07         Start ROM-Nummer
032B  CD 32 03    CALL  0332         ROM ggf. in VL einh., in Tab.
032E  0D          DEC   C            ROM-Nummer erniedrigen
032F  20 FA       JR    NZ,032B      noch nicht null? dann weiter
0331  C9          RET

*********************************  KL INIT BACK
                                   IN : C: ROM-Nummer
                                        DE: Lo RAM
                                        HL: Hi RAM
                                   OUT: C: ROM-Nummer
                                        B: alte Konfiguration
                                        DE: Lo RAM, neu
                                        HL: Hi RAM, neu
0332  79          LD    A,C          ROM-Nummer
0333  FE 08       CP    08           >7?
0335  D0          RET   NC           dann zurück
0336  CD 7E BA    CALL  BA7E         ROM anschalten
0339  3A 00 C0    LD    A,(C000)     ROM-Kennung
033C  E6 03       AND   03           b1,b0 isolieren
033E  3D          DEC   A            b1=0, b0=1?
033F  20 1F       JR    NZ,0360      sonst ROM aus, raus
0341  C5          PUSH  BC           ROM-Nummer retten
0342  CD 06 C0    CALL  C006         ROM anspringen
0345  D5          PUSH  DE           Lo RAM Zeiger retten
0346  23          INC   HL           Zeiger Hi RAM
0347  EB          EX    DE,HL        nach DE
0348  21 AA B1    LD    HL,B1AA      Tabellenanfang
034B  ED 4B A8 B1 LD    BC,(B1A8)    lfd. ROM-Nummer
034F  06 00       LD    B,00
0351  09          ADD   HL,BC          zweimal f. zwei Bytes pro
0352  09          ADD   HL,BC          Eintrag addieren
0353  73          LD    (HL),E         altes Hi RAM
0354  23          INC   HL             in Tabelle
0355  72          LD    (HL),D         setzen
0356  21 FC FF    LD    HL,FFFC      altes Hi RAM +1 -4
0359  19          ADD   HL,DE        ergibt Zeiger auf VL-Eintrag
035A  CD A1 02    CALL  02A1         ROM in VL eintragen
035D  2B          DEC   HL           neuer Hi RAM Zeiger
035E  D1          POP   DE           neuer Lo RAM Zeiger
035F  C1          POP   BC           alte und lfd. ROM-Nummer
0360  C3 8C BA    JP    BA8C         ROM wieder ausschalten

*********************************  Koppeladresse suchen
                                   IN : HL: Zeiger auf PQ-Zähler
                                        DE: gesuchte Adresse
                                   OUT: HL: Adr. d. Koppeladresse
                                        CY:=1, wenn gefunden
0363  7E          LD    A,(HL)       Lo-Byte
```

```
0364  BB           CP     E           =gesuchtem Lo-Byte?
0365  23           INC    HL          Hi-Byte
0366  7E           LD     A,(HL)      laden
0367  2B           DEC    HL          Zeiger wieder zurück
0368  20 03        JR     NZ,036D     ungleich? d. nächsten Eintrag
036A  BA           CP     D           Hi-Bytes auch gleich?
036B  37           SCF                CY:=1 f. evtl. gefunden
036C  C8           RET    Z           wenn gleich, dann raus
036D  B7           OR     A           Listenende?
036E  C8           RET    Z           dann raus, CY:=0 f. nicht gef.
036F  6E           LD     L,(HL)        Zeiger auf nächsten Eintrag
0370  67           LD     H,A           nach HL laden
0371  18 F0        JR     0363        und nächsten Eintrag bearb.

************************************ Add Event
                                    IN : HL: Zeiger auf KA d. Events
                                         DE: Zeiger auf entspr. Chain
0373  EB           EX     DE,HL         Adresse d. Events nach DE
0374  F3           DI
0375  CD 63 03     CALL   0363          Event in Chain suchen
0378  38 06        JR     C,0380      gefunden? dann raus
037A  73           LD     (HL),E        sonst Event
037B  23           INC    HL            an Chain
037C  72           LD     (HL),D        hinten anhängen
037D  13           INC    DE          Zeiger auf Hi-Byte d. KA
037E  AF           XOR    A           Markierung f. Kettenende
037F  12           LD     (DE),A      setzen
0380  FB           EI
0381  C9           RET

************************************ Delete Event
                                    IN : HL: Zeiger auf KA d. Events
                                         DE: Zeiger auf entspr. Chain
                                    OUT: CY:=1, wenn gef. u. gelöscht
                                         DE: Zeiger auf KA+1
                                         HL: Zeiger letzte KA+1
0382  EB           EX     DE,HL         Adresse d. Events nach DE
0383  F3           DI
0384  CD 63 03     CALL   0363          Koppeladresse suchen
0387  30 06        JR     NC,038F     nicht in Chain? dann raus
0389  1A           LD     A,(DE)        sonst KA auf nächsten Event
038A  77           LD     (HL),A        in vorige KA
038B  13           INC    DE            übertragen, d.h.
038C  23           INC    HL            lfd. Event aus Chain
038D  1A           LD     A,(DE)        aushängen
038E  77           LD     (HL),A
038F  FB           EI
0390  C9           RET

************************************ Kernel Hi Jumps
B900 0391  C3 5E BA    JP     BA5E        KL U ROM ENABLE
B903 0394  C3 68 BA    JP     BA68        KL U ROM DISABLE
B906 0397  C3 4A BA    JP     BA4A        KL L ROM ENABLE
B909 039A  C3 54 BA    JP     BA54        KL L ROM DISABLE
B90C 039D  C3 72 BA    JP     BA72        KL ROM RESTORE
B90F 03A0  C3 7E BA    JP     BA7E        KL ROM SELECT
B912 03A3  C3 A2 BA    JP     BAA2        KL CURR SELECTION
B915 03A6  C3 83 BA    JP     BA83        KL PROBE ROM
```

```
B918 03A9  C3 8C BA     JP     BA8C        KL ROM DESELECT
B91B 03AC  C3 A6 BA     JP     BAA6        KL LDIR
B91E 03AF  C3 AC BA     JP     BAAC        KL LDDR

***********************************  KL POLL SYNCHRONOUS
                                     OUT: CY=0, wenn kein Eintrag mit
                                          ausreichend. Priorität
B921 03B2  3A 94 B1     LD     A,(B194)    lfd. sync. Event
B924 03B5  B7           OR     A           kein Eintrag in SPQ?
B925 03B6  C8           RET    Z           dann zurück
B926 03B7  E5           PUSH   HL
B927 03B8  F3           DI
B928 03B9  2A 93 B1     LD     HL,(B193)   Adr. d. lfd. sync. Events
B92B 03BC  7C           LD     A,H         Ende der Pending Queue?
B92C 03BD  B7           OR     A           Ende der Queue?
B92D 03BE  28 07        JR     Z,03C7      dann raus
B92F 03C0  23           INC    HL            Zeiger auf Priorität
B930 03C1  23           INC    HL            d. lfd. Events
B931 03C2  23           INC    HL
B932 03C3  3A 95 B1     LD     A,(B195)    Zeiger auf lfd. Priorität
B935 03C6  BE           CP     (HL)        m. Prior. lfd. Event vergl.
B936 03C7  E1           POP    HL
B937 03C8  FB           EI
B938 03C9  C9           RET

***********************************  INTERRUPT ENTRY CONT'D
B939 03CA  F3           DI
B93A 03CB  08           EX     AF,AF'      Interrupt bereits laufend?
B93B 03CC  38 33        JR     C,0401      dann EXT INT ENTRY
B93D 03CE  D9           EXX                Interrupt-Registersatz ein
B93E 03CF  79           LD     A,C         lfd. System Status (SYSTAT)
B93F 03D0  37           SCF                CY:=1 f. Interruptbeh. lfd.
B940 03D1  FB           EI                 externen Interrupt zulassen
B941 03D2  08           EX     AF,AF'
B942 03D3  F3           DI
B943 03D4  F5           PUSH   AF          Akku retten
B944 03D5  CB 91        RES    2,C         Lo-ROM
B946 03D7  ED 49        OUT    (C),C       anschalten
B948 03D9  CD B1 00     CALL   00B1        Scan Events, Chains bearb.
B94B 03DC  B7           OR     A           CY:=0
B94C 03DD  08           EX     AF,AF'      in Interrupt-Registersatz
B94D 03DE  4F           LD     C,A         alter SYSTAT
B94E 03DF  06 7F        LD     B,7F        ins Gate Array
B950 03E1  3A 04 B1     LD     A,(B104)    Interrupt-Flags
B953 03E4  B7           OR     A           testen
B954 03E5  28 14        JR     Z,03FB      keine APQ-Eintr., TC inakt.
B956 03E7  FA 6A B9     JP     M,B96A      APQ in Bearbeitung? d. raus
B959 03EA  79           LD     A,C         SYSTAT
B95A 03EB  E6 0C        AND    0C          ROM-Switches isolieren
B95C 03ED  F5           PUSH   AF          und retten
B95D 03EE  CB 91        RES    2,C         Hi-ROM anschalten
B95F 03F0  D9           EXX                norm. Registersatz
B960 03F1  CD 0A 01     CALL   010A        APQ u. TC bearbeiten
B963 03F4  D9           EXX                Interrupt-Registersatz
B964 03F5  E1           POP    HL          ROM-Switches
B965 03F6  79           LD     A,C         SYSTAT
B966 03F7  E6 F3        AND    F3            alten ROM-Status
B968 03F9  B4           OR     H             wiederherstellen
```

```
B969 03FA  4F             LD     C,A           und nach C als SYSTAT
B96A 03FB  ED 49          OUT    (C),C         SYSTAT ins GA schreiben
B96C 03FD  D9             EXX                  norm. Registersatz
B96D 03FE  F1             POP    AF            alter Akku
B96E 03FF  FB             EI
B96F 0400  C9             RET

************************************  EXT INTERRUPT ENTRY
B970 0401  08             EX     AF,AF'        alter Akku
B971 0402  E1             POP    HL            RET-Adresse löschen
B972 0403  F5             PUSH   AF            Akku retten
B973 0404  CB D1          SET    2,C           Lo-ROM
B975 0406  ED 49          OUT    (C),C         ausschalten
B977 0408  CD 3B 00       CALL   003B          externen Interrupt ausführ.
B97A 040B  18 CF          JR     03DC          und in Interrupt-Behandlung

************************************  KL LO PCHL CONT'D
                                      IN : HL<13..0>: Sprungadresse
                                           HL<15..14>: Konfiguration
B97C 040D  F3             DI
B97D 040E  E5             PUSH   HL            Sprungadr. u. Konfig.
B97E 040F  D9             EXX                  2. Registersatz
B97F 0410  D1             POP    DE            Sprungadr. nach DE
B980 0411  18 06          JR     0419          Sprung ausführen

************************************  KL LO JUMP CONT'D
                                      IN : (SP): Sprungadr. u. Konfig.
                                      OUT: SP:=SP+2
B982 0413  F3             DI
B983 0414  D9             EXX                  2. Registersatz
B984 0415  E1             POP    HL            RET-Adresse
B985 0416  5E             LD     E,(HL)          Sprungadr. nach Aufruf
B986 0417  23             INC    HL              nach DE
B987 0418  56             LD     D,(HL)
B988 0419  08             EX     AF,AF'        Akku retten
B989 041A  7A             LD     A,D           Konfig. nach A retten
B98A 041B  CB BA          RES    7,D             Konfig. löschen,
B98C 041D  CB B2          RES    6,D             ergibt Sprungadr. in DE
B98E 041F  07             RLCA
B98F 0420  07             RLCA                   Konfig.-Bits in A
B990 0421  07             RLCA                   in die Position b3/b2
B991 0422  07             RLCA                   schieben
B992 0423  A9             XOR    C
B993 0424  E6 0C          AND    0C              und in SYSTAT hinein
B995 0426  A9             XOR    C               setzen
B996 0427  C5             PUSH   BC            alten SYSTAT retten
B997 0428  CD A8 B9       CALL   B9A8          Konfig. setzen, Rout. aufr.
B99A 042B  F3             DI
B99B 042C  D9             EXX                  alten Registersatz wieder
B99C 042D  08             EX     AF,AF'        anschalten
B99D 042E  79             LD     A,C           neue Konfig.
B99E 042F  C1             POP    BC            alte Konfig.
B99F 0430  E6 03          AND    03              neuen Bildschirm-Modus
B9A1 0432  CB 89          RES    1,C             in alten SYSTAT,
B9A3 0434  CB 81          RES    0,C             d.h. beibehalten
B9A5 0436  B1             OR     C
B9A6 0437  18 01          JR     043A          SYSTAT setzen
```

```
*********************************   Sprung nach DE ausführen
                                    IN : DE: Sprungadresse
                                          A: Wert f. Gate Array
B9A8 0439  D5          PUSH   DE          Sprungadr. auf Stack
B9A9 043A  4F          LD     C,A         Byte f. Gate Array
B9AA 043B  ED 49       OUT    (C),C       ins Gate Array schreiben
B9AC 043D  B7          OR     A           CY:=0 f. Interrupt inaktiv
B9AD 043E  08          EX     AF,AF'      ins Interrupt-Carry
B9AE 043F  D9          EXX                norm. Registersatz an
B9AF 0440  FB          EI
B9B0 0441  C9          RET                Routine anspringen

*********************************   KL FAR PCHL
                                    IN : HL: Routinenadresse
                                          C: Konfig. f. Aufruf
B9B1 0442  F3          DI
B9B2 0443  08          EX     AF,AF'      Akku retten
B9B3 0444  79          LD     A,C         Konfig. nach A
B9B4 0445  E5          PUSH   HL          Sprungadresse
B9B5 0446  D9          EXX                1. Registersatz retten
B9B6 0447  D1          POP    DE          Sprungadresse nach DE
B9B7 0448  18 15       JR     045F        Sprung ausführen

*********************************   KL FAR CALL
                                    IN : HL: Zeiger auf Routinenadr.
                                             und Konfiguration
B9B9 044A  F3          DI
B9BA 044B  E5          PUSH   HL
B9BB 044C  D9          EXX
B9BC 044D  E1          POP    HL
B9BD 044E  18 09       JR     0459

*********************************   KL LO FAR CALL CONT'D
                                    IN : ((SP)): Routinenadresse
                                          ((SP)+2): Konfiguration
B9BF 0450  F3          DI
B9C0 0451  D9          EXX                1. Registersatz retten
B9C1 0452  E1          POP    HL          RET-Adresse nach HL
B9C2 0453  5E          LD     E,(HL)        auf Aufruf folgende
B9C3 0454  23          INC    HL            Adresse nach DE
B9C4 0455  56          LD     D,(HL)        laden
B9C5 0456  23          INC    HL          und neue RET-Adresse
B9C6 0457  E5          PUSH   HL          auf Stack
B9C7 0458  EB          EX     DE,HL       Adresse nach Aufruf nach HL
B9C8 0459  5E          LD     E,(HL)        Routinenadresse
B9C9 045A  23          INC    HL            nach DE
B9CA 045B  56          LD     D,(HL)        laden
B9CB 045C  23          INC    HL          Zeiger a. Konfig. f. Aufruf
B9CC 045D  08          EX     AF,AF'      Akku retten
B9CD 045E  7E          LD     A,(HL)      Konfiguration laden

B9CE 045F  FE FC       CP     FC          externes ROM?
B9D0 0461  30 BE       JR     NC,0421     sonst SYSTAT setzen
B9D2 0463  06 DF       LD     B,DF        I/O-Adr. f. ROM-Auswahl
B9D4 0465  ED 79       OUT    (C),A       ROM-Nummer setzen
B9D6 0467  21 A8 B1    LD     HL,B1A8     alte ROM-Nummer
B9D9 046A  46          LD     B,(HL)      laden
B9DA 046B  77          LD     (HL),A      neue setzen
```

```
B9DB 046C  C5           PUSH   BC           alte ROM-Nummer retten
B9DC 046D  FD E5        PUSH   IY
B9DE 046F  3D           DEC    A            ROM-Nummer
B9DF 0470  FE 07        CP     07           von 01..08?
B9E1 0472  30 0F        JR     NC,0483      sonst raus
B9E3 0474  87           ADD    A            ggf. *2, f. 2 Bytes/Eintrag
B9E4 0475  C6 AC        ADD    AC             und zu Tabellenstart
B9E6 0477  6F           LD     L,A            (B1AC) addieren
B9E7 0478  CE B1        ADC    B1             Ergebnis
B9E9 047A  95           SUB    L              nach HL
B9EA 047B  67           LD     H,A
B9EB 047C  7E           LD     A,(HL)         Einsprung f. ROM
B9EC 047D  23           INC    HL             aus der Tabelle
B9ED 047E  66           LD     H,(HL)         holen,
B9EE 047F  6F           LD     L,A            nach HL
B9EF 0480  E5           PUSH   HL           und Einsprung
B9F0 0481  FD E1        POP    IY           nach IY
B9F2 0483  06 7F        LD     B,7F         I/O-Adr. d. Gate Array
B9F4 0485  79           LD     A,C          SYSTAT nach A
B9F5 0486  CB D7        SET    2,A          Lo-ROM ausschalten
B9F7 0488  CB 9F        RES    3,A          Hi-ROM einschalten
B9F9 048A  CD A8 B9     CALL   B9A8         Routine aufrufen
B9FC 048D  FD E1        POP    IY
B9FE 048F  F3           DI
B9FF 0490  D9           EXX                   1. Registersatz wieder an
BA00 0491  08           EX     AF,AF'       alter Akku wieder an
BA01 0492  59           LD     E,C          alter SYSTAT
BA02 0493  C1           POP    BC           alte ROM-Nummer
BA03 0494  78           LD     A,B          nach A
BA04 0495  06 DF        LD     B,DF         I/O-Adr. f. ROM-Auswahl
BA06 0497  ED 79        OUT    (C),A        alte ROM-No. wieder setzen
BA08 0499  32 A8 B1     LD     (B1A8),A     und als alte No. setzen
BA0B 049C  06 7F        LD     B,7F         I/O-Adr. d. Gate Array
BA0D 049E  7B           LD     A,E          alten SYSTAT nach A
BA0E 049F  18 8F        JR     0430         alte Konfig. wieder setzen

*********************************** KL SIDE PCHL CONT'D
                                    IN : HL: Routinenadresse
BA10 04A1  F3           DI
BA11 04A2  E5           PUSH   HL             Adr. d. Routine
BA12 04A3  D9           EXX                   1. Registersatz retten
BA13 04A4  D1           POP    DE           Routinenadresse nach DE
BA14 04A5  18 08        JR     04AF         Routine anspringen

*********************************** KL LO SIDE CALL CONT'D
                                    IN : (SP): Adr. d. Routinenadr.
                                    OUT: SP:=SP+2
BA16 04A7  F3           DI
BA17 04A8  D9           EXX                   1. Registersatz retten
BA18 04A9  E1           POP    HL           RET-Adresse vom Stack
BA19 04AA  5E           LD     E,(HL)         Adresse nach Aufruf
BA1A 04AB  23           INC    HL             nach DE laden
BA1B 04AC  56           LD     D,(HL)
BA1C 04AD  23           INC    HL           neue RET-Adresse
BA1D 04AE  E5           PUSH   HL           auf Stack
BA1E 04AF  08           EX     AF,AF'       Akku retten
BA1F 04B0  7A           LD     A,D          Hi-Byte d. Adresse
BA20 04B1  CB FA        SET    7,D            Bit 14 u. 15 setzen f.
```

```
BA22 04B3  CB F2        SET    6,D            Sprung ins Hi-ROM
BA24 04B5  E6 C0        AND    C0           ROM-Select-Bits isolieren
BA26 04B7  07           RLCA                und in b1,b0
BA27 04B8  07           RLCA                schieben
BA28 04B9  21 AB B1     LD     HL,B1AB      Konfig. b. ROM-Einsprung
BA2B 04BC  86           ADD    (HL)         addieren
BA2C 04BD  18 A4        JR     0463         und Sprung ausführen

***********************************   KL FIRM JUMP CONT'D
                                      IN : ((SP)): Routinenadr.
BA2E 04BF  F3           DI
BA2F 04C0  D9           EXX                 1. Registersatz retten
BA30 04C1  E1           POP    HL           Aufrufadresse
BA31 04C2  5E           LD     E,(HL)         Adresse nach Aufruf
BA32 04C3  23           INC    HL             nach DE
BA33 04C4  56           LD     D,(HL)         laden
BA34 04C5  CB 91        RES    2,C          Lo-ROM
BA36 04C7  ED 49        OUT    (C),C        einschalten
BA38 04C9  ED 53 3F BA  LD     (BA3F),DE    Adr. f. CALL speichern
BA3C 04CD  D9           EXX                 1. Registersatz wieder ein
BA3D 04CE  FB           EI
BA3E 04CF  CD 3E BA     CALL   BA3E         Routine aufrufen
BA41 04D2  F3           DI
BA42 04D3  D9           EXX                 2. Registersatz einschalten
BA43 04D4  CB D1        SET    2,C          Lo-ROM wieder
BA45 04D6  ED 49        OUT    (C),C        ausschalten
BA47 04D8  D9           EXX                 1. Registersatz
BA48 04D9  FB           EI
BA49 04DA  C9           RET

***********************************   KL L ROM ENABLE
BA4A 04DB  F3           DI
BA4B 04DC  D9           EXX                 2. Registersatz ein
BA4C 04DD  79           LD     A,C          SYSTAT nach A retten
BA4D 04DE  CB 91        RES    2,C          Lo-ROM
BA4F 04E0  ED 49        OUT    (C),C        einschalten
BA51 04E2  D9           EXX                 1. Registersatz ein
BA52 04E3  FB           EI
BA53 04E4  C9           RET

***********************************   KL L ROM DISABLE
BA54 04E5  F3           DI
BA55 04E6  D9           EXX                 2. Registersatz ein
BA56 04E7  79           LD     A,C          SYSTAT nach A retten
BA57 04E8  CB D1        SET    2,C          Lo-ROM
BA59 04EA  ED 49        OUT    (C),C        ausschalten
BA5B 04EC  D9           EXX                 1. Registersatz ein
BA5C 04ED  FB           EI
BA5D 04EE  C9           RET

***********************************   KL U ROM ENABLE
BA5E 04EF  F3           DI
BA5F 04F0  D9           EXX                 2. Registersatz ein
BA60 04F1  79           LD     A,C          SYSTAT nach A retten
BA61 04F2  CB 99        RES    3,C          Hi-ROM
BA63 04F4  ED 49        OUT    (C),C        einschalten
BA65 04F6  D9           EXX                 1. Registersatz ein
BA66 04F7  FB           EI
```

```
BA67 04F8  C9          RET

*********************************** KL U ROM DISABLE
BA68 04F9  F3          DI
BA69 04FA  D9          EXX                 2. Registersatz ein
BA6A 04FB  79          LD    A,C           SYSTAT nach A retten
BA6B 04FC  CB D9       SET   3,C           Hi-ROM
BA6D 04FE  ED 49       OUT   (C),C         ausschalten
BA6F 0500  D9          EXX                 1. Registersatz ein
BA70 0501  FB          EI
BA71 0502  C9          RET

*********************************** KL ROM RESTORE
BA72 0503  F3          DI
BA73 0504  D9          EXX                 2. Registersatz ein
BA74 0505  A9          XOR   C               b3,b2 d. Akkus
BA75 0506  E6 0C       AND   0C              in lfd. SYSTAT, d.h.
BA77 0508  A9          XOR   C               alte ROM-Setzung
BA78 0509  4F          LD    C,A             wieder
BA79 050A  ED 49       OUT   (C),C         einschalten
BA7B 050C  D9          EXX                 1. Registersatz ein
BA7C 050D  FB          EI
BA7D 050E  C9          RET

*********************************** KL ROM SELECT
                                   IN : C: ROM-Nummer
BA7E 050F  CD 5E BA    CALL  BA5E          Hi-ROM einschalten
BA81 0512  18 0F       JR    0523          ROM auswählen

*********************************** KL PROBE ROM
                                   IN : C: ROM-Nummer
                                   OUT: A: ROM-Kennung
                                        HL: Marke u. Versionsnummer
BA83 0514  CD 7E BA    CALL  BA7E          ROM anschalten
BA86 0517  3A 00 C0    LD    A,(C000)      ROM-Kennung laden
BA89 051A  2A 01 C0    LD    HL,(C001)     Marke u. Versionsno. laden

*********************************** KL ROM DESELECT
                                   IN : B<3..2>: neue ROM-Bits
                                        C: neue ROM-Nummer
                                        A: alte ROM-Bits
                                   OUT: A,B: alte ROM-Bits
                                        C: alte ROM-Nummer
BA8C 051D  F5          PUSH  AF            ROM-Kennung retten
BA8D 051E  78          LD    A,B           alte ROM-Bits
BA8E 051F  CD 72 BA    CALL  BA72          wieder setzen
BA91 0522  F1          POP   AF            Akku wieder vom Stack

*********************************** ROM-Nummer schreiben
                                   IN : A: alte ROM-Bits
                                        C: neue ROM-Nummer
                                   OUT: A,B: alte Konfig.
                                        C: alte ROM-Nummer
BA92 0523  E5          PUSH  HL
BA93 0524  F3          DI
BA94 0525  06 DF       LD    B,DF          I/O-Adr. f. ROM-Auswahl
BA96 0527  ED 49       OUT   (C),C         ROM-Nummer schreiben
BA98 0529  21 A8 B1    LD    HL,B1A8       alte ROM-Nummer
```

```
BA9B 052C  46           LD     B,(HL)       laden
BA9C 052D  71           LD     (HL),C       neue ROM-Nummer setzen
BA9D 052E  48           LD     C,B          alte ROM-Nummer nach C
BA9E 052F  47           LD     B,A          alte Konfig. nach B
BA9F 0530  FB           EI
BAA0 0531  E1           POP    HL
BAA1 0532  C9           RET

***********************************  KL CURR SELECTION
                                     OUT: A: lfd. ROM-Nummer
BAA2 0533  3A A8 B1     LD     A,(B1A8)     lfd. ROM-Nummer laden
BAA5 0536  C9           RET

***********************************  KL LDIR
                                     IN : HL: Zeiger auf Quellblock
                                          DE: Zeiger auf Zielblock
                                          BC: Länge
                                     OUT: HL: Zeiger nach Quellblock
                                          DE: Zeiger nach Zielblock
                                          BC: immer 0
                                          V=0, N=0, H=0
BAA6 0537  CD B2 BA     CALL   BAB2          ROMs vorübergeh. abschalten
BAA9 053A  ED B0        LDIR                 Block kopieren
BAAB 053C  C9           RET

***********************************  KL LDDR
                                     IN : HL: Zeiger a. Quellblockende
                                          DE: Zeiger auf Zielblockende
                                          BC: Länge
                                     OUT: HL: Zeiger vor Quellblock
                                          DE: Zeiger vor Zielblock
                                          BC: immer 0
                                          V=0, N=0, H=0
BAAC 053D  CD B2 BA     CALL   BAB2          ROMs vorübergeh. abschalten
BAAF 0540  ED B8        LDDR                 Block kopieren
BAB1 0542  C9           RET

***********************************  ROMs transparent abschalten
BAB2 0543  F3           DI
BAB3 0544  D9           EXX                  2. Registersatz ein
BAB4 0545  E1           POP    HL            RET-Adresse vom Stack
BAB5 0546  C5           PUSH   BC            alten SYSTAT retten
BAB6 0547  CB D1        SET    2,C             beide ROMs
BAB8 0549  CB D9        SET    3,C             abschalten
BABA 054B  ED 49        OUT    (C),C         neuen SYSTAT ausgeben
BABC 054D  CD C7 BA     CALL   BAC7          restl. Routine erst ausf.
BABF 0550  F3           DI                   nach Beendigung derselben:
BAC0 0551  D9           EXX                  2. Registersatz
BAC1 0552  C1           POP    BC            alten SYSTAT
BAC2 0553  ED 49        OUT    (C),C         wiederherstellen
BAC4 0555  D9           EXX                  1. Registersatz ein
BAC5 0556  FB           EI
BAC6 0557  C9           RET

BAC7 0558  E5           PUSH   HL            Adr. d. aufruf. Routine
BAC8 0559  D9           EXX                  1. Registersatz ein
BAC9 055A  FB           EI
BACA 055B  C9           RET                  Routine anspringen
```

```
***********************************   KL RAM LAM,  HL
                                      (läßt Flags unverändert)
                                      IN : HL: Adresse
                                      OUT: A: Byte von (HL)
BACB 055C  F3          DI
BACC 055D  D9          EXX                  2. Registersatz ein
BACD 055E  59          LD     E,C           SYSTAT nach E kopieren
BACE 055F  CB D3       SET    2,E             alle 64KB RAM
BAD0 0561  CB DB       SET    3,E             einschalten
BAD2 0563  ED 59       OUT    (C),E         neuen SYSTAT ausgeben
BAD4 0565  D9          EXX                  wieder 1. Registersatz ein
BAD5 0566  7E          LD     A,(HL)        Byte aus RAM laden
BAD6 0567  D9          EXX                  2. Registersatz ein
BAD7 0568  ED 49       OUT    (C),C         wieder alten SYSTAT setzen
BAD9 056A  D9          EXX                  1. Registersatz ein
BADA 056B  FB          EI
BADB 056C  C9          RET

***********************************   KL RAM LAM,  IX
                                      IN : IX: Adresse
                                      OUT: A: Byte von (IX)
BADC 056D  D9          EXX                  2. Registersatz
BADD 056E  79          LD     A,C           lfd. SYSTAT nach A
BADE 056F  F6 0C       OR     0C            beide ROMs
BAE0 0571  ED 79       OUT    (C),A         ausschalten
BAE2 0573  DD 7E 00    LD     A,(IX+00)     Byte aus RAM laden
BAE5 0576  ED 49       OUT    (C),C         alten SYSTAT wieder setzen
BAE7 0578  D9          EXX                  1. Registersatz wieder ein
BAE8 0579  C9          RET

057A  C7          RST    00
057B  C7          RST    00
057C  C7          RST    00
057D  C7          RST    00
057E  C7          RST    00
057F  C7          RST    00

------------------------ MACHINE PACK (MC) ------------------------

***********************************   RESET CONT'D
0580  F3          DI
0581  01 82 F7    LD     BC,F782      Kontrollregister PIO 8255
0584  ED 49       OUT    (C),C        initialisieren
0586  01 00 F4    LD     BC,F400      $00 aus Port A
0589  ED 49       OUT    (C),C        senden
058B  01 00 F6    LD     BC,F600      $00 aus Port C
058E  ED 49       OUT    (C),C        senden
0590  01 7F EF    LD     BC,EF7F      $7F an den Drucker
0593  ED 49       OUT    (C),C        schicken
0595  06 F5       LD     B,F5         Port B, PIO
0597  ED 78       IN     A,(C)        laden
0599  E6 10       AND    10           Bit f. 50/60Hz Bildwiederh.
059B  21 C4 05    LD     HL,05C4      Tabelle f. 50Hz
059E  20 03       JR     NZ,05A3      wenn =1, d.h. 50Hz
05A0  21 D4 05    LD     HL,05D4      sonst Tabelle f. 60Hz
05A3  01 0F BC    LD     BC,BC0F      Adreßregister d. CRTC
05A6  ED 49       OUT    (C),C        Adresse in CRTC schreiben
05A8  2B          DEC    HL           nächstes Byte aus Tabelle
```

```
05A9  7E          LD    A,(HL)       laden
05AA  04          INC   B            und ins entspr. CRTC-Register
05AB  ED 79       OUT   (C),A        schreiben
05AD  05          DEC   B            CRTC-Adreßregister
05AE  0D          DEC   C            Registernummer erniedrigen
05AF  F2 A6 05    JP    P,05A6       bis alle CRTC-Register gesetzt
05B2  18 20       JR    05D4         Tabellen überspringen

*********************************** CRTC-Werte, 50 Hz
05B4  3F 28 2E 8E 26 00 19 1E
05BC  00 07 00 00 30 00 C0 00

*********************************** CRTC-Werte, 60 Hz
05C4  3F 28 2E 8E 1F 06 19 1B
05CC  00 07 00 00 30 00 C0 00

05D4  11 5C 06    LD    DE,065C       Zeiger auf Einschaltmeldung
05D7  21 00 00    LD    HL,0000       Kennz. f. Basic anspringen
05DA  18 32       JR    060E          System starten

*********************************** MC BOOT PROGRAM
                                   IN : HL: Aufrufadr. d. Programms
05DC  31 00 C0    LD    SP,C000       Stack Startwert
05DF  E5          PUSH  HL            Aufrufadr. retten
05E0  CD 68 1E    CALL  1E68          SOUND RESET
05E3  F3          DI
05E4  01 FF F8    LD    BC,F8FF       $FF an evtl. angeschl. Periph.
05E7  ED 49       OUT   (C),C         senden
05E9  CD 5C 00    CALL  005C          Kernel initialisieren
05EC  E1          POP   HL            Aufrufadresse
05ED  D5          PUSH  DE            Einsprung lfd. ROM
05EE  C5          PUSH  BC            alte/neue ROM-Konfig.
05EF  E5          PUSH  HL            Aufrufadr. d. Programms
05F0  CD 1E 1A    CALL  1A1E          KM RESET
05F3  CD 88 10    CALL  1088          TXT RESET
05F6  CD B1 0A    CALL  0AB1          SCR RESET
05F9  CD 5E BA    CALL  BA5E          Hi-ROM einschalten
05FC  E1          POP   HL            Aufrufadr. d. Boot-Programms
05FD  CD 75 07    CALL  0775          Lade-Routine ausführen
0600  C1          POP   BC            alte/neue ROM-Konfig.
0601  D1          POP   DE            Einsprung in lfd. ROM
0602  38 07       JR    C,060B        kein Fehler? d. System starten
0604  EB          EX    DE,HL         Einsprung in lfd. ROM
0605  48          LD    C,B           alte ROM-Konfig.
0606  11 E8 06    LD    DE,06E8       Adr. f. "PROGRAM LOAD FAILED"
0609  18 03       JR    060E          und System starten

*********************************** MC START PROGRAM
                                   IN : DE: Adr. f. Meldung
                                        HL: Adr. d. Programms
                                          (=$0000 f. Basic)
                                        C: ROM-Konfiguration
060B  11 26 07    LD    DE,0726       Zeiger auf RET (keine Meldung)

060E  F3          DI
060F  ED 56       IM    1
0611  D9          EXX                 2. Registersatz ein
0612  01 00 DF    LD    BC,DF00       $00 als ROM-Nummer
```

```
0615  ED 49        OUT   (C),C        setzen
0617  01 FF F8     LD    BC,F8FF      evtl. Peripherie $FF
061A  ED 49        OUT   (C),C        übergeben
061C  21 00 B1     LD    HL,B100        den Bereich von $B100
061F  11 01 B1     LD    DE,B101        bis $B8FF, d.h.
0622  01 FF 07     LD    BC,07FF        den Arbeitsspeicher
0625  36 00        LD    (HL),00        für das Operating System
0627  ED B0        LDIR                 löschen
0629  01 89 7F     LD    BC,7F89      Hi-ROM aus, Lo-ROM an, Mode 1
062C  ED 49        OUT   (C),C        setzen
062E  D9           EXX                wieder 1. Registersatz
062F  AF           XOR   A            CY:=0 f. nicht im Interrupt
0630  08           EX    AF,AF'       ins Interrupt-Carry
0631  31 00 C0     LD    SP,C000      Stack Startwert
0634  E5           PUSH  HL           Einsrungadresse
0635  C5           PUSH  BC           ROM-Konfiguration
0636  D5           PUSH  DE           Adresse der Meldung
0637  CD 44 00     CALL  0044         Hi Kernel Jumps kopieren
063A  CD 88 08     CALL  0888         Jump Restore
063D  CD E0 19     CALL  19E0         KM Init
0640  CD 68 1E     CALL  1E68         Sound Reset
0643  CD A0 0A     CALL  0AA0         SCR Init
0646  CD 78 10     CALL  1078         TXT Init
0649  CD B0 15     CALL  15B0         GRA Init
064C  CD 70 23     CALL  2370         CAS Init
064F  CD E6 07     CALL  07E6         MC Reset Printer
0652  FB           EI
0653  E1           POP   HL           Adresse f. Meldung
0654  CD 75 07     CALL  0775         Meldung ausgeben
0657  C1           POP   BC           ROM-Konfiguration
0658  E1           POP   HL           Einsprungadresse
0659  C3 77 00     JP    0077         Programm/Basic anspringen

*********************************** Einschalt-Meldung ausgeben
065C  CD 12 07     CALL  0712           Firmennamen-Adresse holen
065F  CD EB 06     CALL  06EB           Firmennamen ausgeben
0662  21 6D 06     LD    HL,066D      "64 K Microcomputer ..."
0665  CD EB 06     CALL  06EB         ausgeben
0668  21 93 06     LD    HL,0693      "(c) 1984 Amstrad Consumer..."
066B  18 7E        JR    06EB         ausgeben

*********************************** Einschaltmeldung
066D  20 36 34 4B 20 4D 69 63          64K Mic
0675  72 6F 63 6F 6D 70 75 74         rocomput
067D  65 72 20 20 28 76 31 29         er  (v1)
0685  0D 0A 0D 0A 00 43 6F 70               Cop
068D  79 72 69 67 68 74 20 A4         yright
0695  31 39 38 34 20 41 6D 73         1984 Ams
069D  74 72 61 64 20 43 6F 6E         trad Con
06A5  73 75 6D 65 72 20 45 6C         sumer El
06AD  65 63 74 72 6F 6E 69 63         ectronic
06B5  73 20 70 6C 63 0D 0A 20         s plc
06BD  20 20 20 20 20 20 20 20
06C5  20 20 61 6E 64 20 4C 6F           and Lo
06CD  63 6F 6D 6F 74 69 76 65         comotive
06D5  20 53 6F 66 74 77 61 72          Softwar
06DD  65 20 4C 74 64 2E 0D 0A         e Ltd.
06E5  0D 0A 00
```

```
**********************************   Ladefehler-Meldung ausgeben
06E8  21 F4 06     LD     HL,06F4       Zeiger auf Ladefehler-Meldung

**********************************   Meldung ausgeben
                                     IN : HL: Adr. d. Meldung
06EB  7E           LD     A,(HL)        Zeichen aus Meldung
06EC  23           INC    HL            Zeiger auf nächstes Zeichen
06ED  B7           OR     A             Ende der Meldung?
06EE  C8           RET    Z             dann raus
06EF  CD 00 14     CALL   1400          Zeichen auf Bildschirm ausgeb.
06F2  18 F7        JR     06EB          und nächstes Zeichen bearb.

**********************************   Ladefehler-Meldung
06F4  2A 2A 2A 20 50 52 4F 47           *** PROG
06FC  52 41 4D 20 4C 4F 41 44           RAM LOAD
0704  20 46 41 49 4C 45 44 20            FAILED
070C  2A 2A 2A 0D 0A 00                 ***

**********************************   Firmennamen-Adresse holen
                                     OUT: HL: Adr. d. Firmennamens
0712  06 F5        LD     B,F5          Port B, PIO
0714  ED 78        IN     A,(C)         laden
0716  2F           CPL                    Bits 1-4
0717  E6 0E        AND    0E              isolieren
0719  0F           RRCA                 und nach unten schieben
071A  21 27 07     LD     HL,0727       Anfang d. Namenstabelle
071D  3C           INC    A             Namensnummer+1
071E  47           LD     B,A           nach B als Zähler
071F  7E           LD     A,(HL)        Zeichen aus Namen
0720  23           INC    HL            Zeiger auf nächstes Zeichen
0721  B7           OR     A             Namensende?
0722  20 FB        JR     NZ,071F       sonst weiter lesen
0724  10 F9        DJNZ   071F          bis Nummer gleich null
0726  C9           RET

**********************************   Firmennamen
0727  41 72 6E 6F 6C 64 00 0A           Arnold
072F  20 41 6D 73 74 72 61 64            Amstrad
0737  00 0A 20 4F 72 69 6F 6E              Orion
073F  00 0A 20 53 63 68 6E 65              Schne
0747  69 64 65 72 00 0A 20 41           ider   A
074F  77 61 00 0A 20 53 6F 6C           wa   Sol
0757  61 76 6F 78 00 0A 20 53           avox   S
075F  61 69 73 68 6F 00 0A 20           aisho
0767  54 72 69 75 6D 70 68 00           Triumph
076F  0A 20 49 73 70 00                   Isp

**********************************   JP (HL)
0775  E9           JP     (HL)

**********************************   MC SET MODE
                                     IN : A: Mode-Nummer
0776  FE 03        CP     03            Mode >2?
0778  D0           RET    NC            dann zurück
0779  F3           DI
077A  D9           EXX                  2. Registersatz an
077B  CB 89        RES    1,C             beide Mode-Bits in
077D  CB 81        RES    0,C             SYSTAT zurücksetzen
```

```
077F  B1          OR     C          und mit neuen Mode-Bits verkn.
0780  4F          LD     C,A        neuen SYSTAT
0781  ED 49       OUT    (C),C      ins GA setzen
0783  FB          EI
0784  D9          EXX               1. Registersatz wieder an
0785  C9          RET

*********************************** MC CLEAR INKS
                                    IN : DE: Adr. d. Farben
0786  C5          PUSH   BC
0787  D5          PUSH   DE         Adr. d. Farben retten
0788  01 10 7F    LD     BC,7F10    GA: Farbstift-Register
078B  CD AB 07    CALL   07AB       Border-Wert übergeben
078E  0E 00       LD     C,00       Farbstift 0, GA Register
0790  CD AB 07    CALL   07AB       Wert übergeben
0793  1B          DEC    DE         Zeiger wieder auf gleich. Wert
0794  20 FA       JR     NZ,0790    schon alle Farben?
0796  D1          POP    DE
0797  C1          POP    BC
0798  C9          RET

*********************************** MC SET INKS
                                    IN : Adr. d. Farbtabelle
0799  C5          PUSH   BC
079A  D5          PUSH   DE
079B  01 10 7F    LD     BC,7F10    GA, Farbstift-Reg., Border
079E  CD AB 07    CALL   07AB       Wert übergeben
07A1  0E 00       LD     C,00       GA, Farbstift-Reg., Stift 0
07A3  CD AB 07    CALL   07AB       Wert übergeben
07A6  20 FB       JR     NZ,07A3    schon alle Stifte?
07A8  D1          POP    DE
07A9  C1          POP    BC
07AA  C9          RET

*********************************** Farbwert in Gate Array schreiben
                                    IN : C: Farbstift-Nummer
                                         DE: Tabellenzeiger
                                    OUT: C: nächste Farbstift-No.
                                         DE: Zeiger auf nächsten Wert
                                         Z:=1, wenn alle 16 Stifte
07AB  ED 49       OUT    (C),C      Farbregister-Nummer ausgeben
07AD  1A          LD     A,(DE)     Farbwert aus Tabelle
07AE  13          INC    DE         Zeiger auf nächsten Tab.-Wert
07AF  E6 1F       AND    1F           Farbwertregister
07B1  F6 40       OR     40           auswählen
07B3  ED 79       OUT    (C),A      und Farbwert schreiben
07B5  0C          INC    C          nächste Farbstift-Nummer
07B6  79          LD     A,C        nach A f. Vergleich
07B7  FE 10       CP     10         schon alle 16 Farbstifte?
07B9  C9          RET

*********************************** MC WAIT FLYBACK
07BA  F5          PUSH   AF
07BB  C5          PUSH   BC
07BC  06 F5       LD     B,F5       Port B, PIO
07BE  ED 78       IN     A,(C)      laden
07C0  1F          RRA               VSYNC ins Carry schieben
07C1  30 FB       JR     NC,07BE    nicht gesetzt? d. warten
```

```
07C3  C1          POP    BC
07C4  F1          POP    AF
07C5  C9          RET

***********************************  MC SCREEN OFFSET
                                     IN : A: SCR BASE, in 16K Blöcken
                                          HL: SCR OFFSET
07C6  C5          PUSH   BC
07C7  0F          RRCA                    SCR BASE nach b5,b4
07C8  0F          RRCA                    schieben
07C9  E6 30       AND    30             signifik. Bits isolieren
07CB  4F          LD     C,A            Ergebnis nach C
07CC  7C          LD     A,H            SCR OFFSET, Hi
07CD  1F          RRA                   wegen Adreßversch. nach unten
07CE  E6 03       AND    03             RA-Bits:=0, oberste Zeichenzl.
07D0  B1          OR     C              m. SCR BASE verknüpfen
07D1  01 0C BC    LD     BC,BC0C        CRTC-Adreßregister, Reg. 12
07D4  ED 49       OUT    (C),C          auswählen
07D6  04          INC    B              CRTC-Datenregister
07D7  ED 79       OUT    (C),A          Startadresse Hi übergeben
07D9  05          DEC    B              CRTC-Adreßregister
07DA  0C          INC    C              Register 13
07DB  ED 49       OUT    (C),C          auswählen
07DD  04          INC    B              CRTC-Datenregister
07DE  7C          LD     A,H              durch 2 teilen,
07DF  1F          RRA                     wegen Adreßverschiebung
07E0  7D          LD     A,L
07E1  1F          RRA
07E2  ED 79       OUT    (C),A          Startadresse lo übergeben
07E4  C1          POP    BC
07E5  C9          RET

***********************************  MC RESET PRINTER
07E6  21 EC 07    LD     HL,07EC        Zeiger Indir. f. WAIT PRINTER
07E9  C3 8A 0A    JP     0A8A           Indirection kopieren

***********************************  Indirection f. MC WAIT PRINTER
07EC  03                                Länge des Blocks
07ED  F1 BD                             Zieladresse
07EF  C3 F8 07    JP     07F8           Indirection

***********************************  MC PRINT CHAR
                                     IN : A: Zeichen
                                     OUT: CY:=1, wenn o.k.
                                          CY:=0, wenn busy
07F2  C5          PUSH   BC
07F3  CD F1 BD    CALL   BDF1           Zeichen ausgeben
07F6  C1          POP    BC
07F7  C9          RET

***********************************  MC WAIT PRINTER
                                     IN : A: Zeichen
                                     OUT: CY:=1, wenn o.k.
                                          CY:=0, wenn busy
07F8  01 32 00    LD     BC,0032        Wartezähler ($3200)
07FB  CD 1B 08    CALL   081B           Printer busy?
07FE  30 07       JR     NC,0807        nein? d. Zeichen an Printer
```

```
0800  10 F9     DJNZ   07FB       weiter warten
0802  0D        DEC    C          Hi-Byte (logisches Hi-Byte)
0803  20 F6     JR     NZ,07FB    und ggf. weiter warten
0805  B7        OR     A          CY:=0 f. busy
0806  C9        RET

*********************************  MC SEND PRINTER
                                   IN : A: Zeichen
                                   OUT: CY:=1, f. fehlerfrei
0807  C5        PUSH   BC
0808  06 EF     LD     B,EF         I/O-Adresse f. Printer
080A  E6 7F     AND    7F           Strobe löschen
080C  ED 79     OUT    (C),A        und Zeichen ausgeben
080E  F6 80     OR     80           Strobe setzen
0810  F3        DI
0811  ED 79     OUT    (C),A        und ausgeben
0813  E6 7F     AND    7F           Strobe wieder löschen
0815  FB        EI
0816  ED 79     OUT    (C),A        und ausgeben
0818  C1        POP    BC
0819  37        SCF                 CY:=1 f. o.k.
081A  C9        RET

*********************************  MC BUSY PRINTER
                                   OUT: CY:=1, wenn Printer busy
081B  C5        PUSH   BC
081C  4F        LD     C,A          retten
081D  06 F5     LD     B,F5         Port B, PIO
081F  ED 78     IN     A,(C)        laden
0821  17        RLA                   Busy-Bit ins Carry
0822  17        RLA                   schieben
0823  79        LD     A,C
0824  C1        POP    BC
0825  C9        RET

*********************************  MC SOUND REGISTER
                                   IN : A: Register-Nummer
                                        C: Daten-Byte
0826  F3        DI
0827  06 F4     LD     B,F4         Port A, PIO
0829  ED 79     OUT    (C),A        Register-Nummer an PSG
082B  06 F6     LD     B,F6         Port C, PIO
082D  ED 78     IN     A,(C)        laden
082F  F6 C0     OR     C0           PSG auf 'Reg.-No. latchen'
0831  ED 79     OUT    (C),A        setzen
0833  E6 3F     AND    3F           PSG auf 'inaktiv'
0835  ED 79     OUT    (C),A        setzen
0837  06 F4     LD     B,F4         Port A, PIO
0839  ED 49     OUT    (C),C        Daten-Byte an PSG
083B  06 F6     LD     B,F6         Port C, PIO
083D  4F        LD     C,A          alter Port C-Wert nach C
083E  F6 80     OR     80           PSG auf 'write'
0840  ED 79     OUT    (C),A        setzen, Daten übernehmen
0842  ED 49     OUT    (C),C        und PSG wieder inaktiv
0844  FB        EI
0845  C9        RET
```

```
************************************     Scan Keyboard
                                         IN : HL: Adr. prim. Rückm.
                                              DE: Adr. pos. Rückm.
                                         OUT: neue Tabellen
0846  01 0E F4     LD     BC,F40E           Port A, PSG-Reg. No. 14
0849  ED 49        OUT    (C),C             auswählen
084B  06 F6        LD     B,F6              Motor & WR DATA
084D  ED 78        IN     A,(C)             laden
084F  E6 30        AND    30                PSG inaktiv, Zeile 0
0851  4F           LD     C,A               Byte retten
0852  F6 C0        OR     C0                Reg.-No. in PSG
0854  ED 79        OUT    (C),A             übernehmen
0856  ED 49        OUT    (C),C             und PSG wieder 'inaktiv'
0858  04           INC    B                 Steuerregister
0859  3E 92        LD     A,92              A,B: Eingang, C: Ausgang
085B  ED 79        OUT    (C),A             setzen
085D  C5           PUSH   BC                Steuerreg.-Adr. retten
085E  CB F1        SET    6,C               von PSG lesen
0860  06 F6        LD     B,F6              Port C, PIO
0862  ED 49        OUT    (C),C             auf 'lesen' u. Tastaturzeile
0864  06 F4        LD     B,F4              Rückmeldung v. Tastatur
0866  ED 78        IN     A,(C)             lesen
0868  46           LD     B,(HL)            letzter Wert b. Tastaturabfr.
0869  77           LD     (HL),A            Rückmeldung in Tabelle
086A  A0           AND    B                   m. letzter Rückmeldung
086B  2F           CPL                        verknüpfen und
086C  12           LD     (DE),A            positiv abspeichern
086D  23           INC    HL                Zeiger auf nächstes Tab.-Byte
086E  13           INC    DE                Zeiger auf nächstes Tab.-Byte
086F  0C           INC    C                 nächste Tastaturzeile
0870  79           LD     A,C               nach A f. Vergleich
0871  E6 0F        AND    0F                schon
0873  FE 0A        CP     0A                die zehnte Zeile?
0875  20 E9        JR     NZ,0860           sonst nächste Zeile abfragen
0877  C1           POP    BC                Adr. d. Steuerregisters
0878  3E 82        LD     A,82              A,C: Ausgang, B: Eingang
087A  ED 79        OUT    (C),A             setzen
087C  05           DEC    B                 Port C
087D  ED 49        OUT    (C),C             PSG 'inaktiv', Zeile 0
087F  C9           RET

0880  C7           RST    00
0881  C7           RST    00
0882  C7           RST    00
0883  C7           RST    00
0884  C7           RST    00
0885  C7           RST    00
0886  C7           RST    00
0887  C7           RST    00
```

```
-------------------------- JUMP RESTORE --------------------------

***********************************  JUMP RESTORE
0888  11 AC 08    LD     DE,08AC      Adresse der Default-Adressentabelle
088B  21 00 BB    LD     HL,BB00      Adresse der RAM-Sprungtabelle
088E  01 CF BF    LD     BC,BFCF      191 Sprünge mit RST 08
0891  CD 97 08    CALL   0897         Ansprünge kopieren
0894  01 EF 30    LD     BC,30EF      48 Sprünge mit RST 28
0897  71          LD     (HL),C       RST-Befehl setzen
0898  23          INC    HL
0899  1A          LD     A,(DE)       Ansprung-Adresse lo
089A  77          LD     (HL),A       kopieren
089B  13          INC    DE
089C  23          INC    HL
089D  EB          EX     DE,HL
089E  79          LD     A,C          RST-Opcode
089F  2F          CPL                   b7 gesetzt, wenn RST 08
08A0  07          RLCA                  (für ROM-Konfiguration,
08A1  07          RLCA                  oberes ROM aus, unteres ein)
08A2  E6 80       AND    80           ROM-Konfig.-Bit isolieren
08A4  B6          OR     (HL)         und in Adresse hi setzen
08A5  EB          EX     DE,HL
08A6  77          LD     (HL),A       Ansprung-Adresse hi speichern
08A7  13          INC    DE
08A8  23          INC    HL
08A9  10 EC       DJNZ   0897         weitere Sprungvektoren ?
08AB  C9          RET

***********************************  Default-Adressentabelle
BB00  CF E0 99   08AC  E0 19  RST 08  19E0   KM INITIALIZE
BB03  CF 1E 9A   08AE  1E 1A  RST 08  1A1E   KM RESET
BB06  CF 3C 9A   08B0  3C 1A  RST 08  1A3C   KM WAIT CHAR
BB09  CF 42 9A   08B2  42 1A  RST 08  1A42   KM READ CHAR
BB0C  CF 77 9A   08B4  77 1A  RST 08  1A77   KM CHAR RETURN
BB0F  CF BD 9A   08B6  BD 1A  RST 08  1ABD   KM SET EXPAND
BB12  CF 2E 9B   08B8  2E 1B  RST 08  1B2E   KM GET EXPAND
BB15  CF 7B 9A   08BA  7B 1A  RST 08  1A7B   KM EXP BUFFER RESET
BB18  CF 56 9B   08BC  56 1B  RST 08  1B56   KM WAIT KEY
BB1B  CF 5C 9B   08BE  5C 1B  RST 08  1B5C   KM READ KEY
BB1E  CF BD 9C   08C0  BD 1C  RST 08  1CBD   KM TEST KEY
BB21  CF B3 9B   08C2  B3 1B  RST 08  1BB3   KM GET STATE
BB24  CF 5C 9C   08C4  5C 1C  RST 08  1C5C   KM GET JOYSTICK
BB27  CF 52 9D   08C6  52 1D  RST 08  1D52   KM SET TRANSLATE
BB2A  CF 3E 9D   08C8  3E 1D  RST 08  1D3E   KM GET TRANSLATE
BB2D  CF 57 9D   08CA  57 1D  RST 08  1D57   KM SET SHIFT
BB30  CF 43 9D   08CC  43 1D  RST 08  1D43   KM GET SHIFT
BB33  CF 5C 9D   08CE  5C 1D  RST 08  1D5C   KM SET CTRL
BB36  CF 48 9D   08D0  48 1D  RST 08  1D48   KM GET CTRL
BB39  CF AB 9C   08D2  AB 1C  RST 08  1CAB   KM SET REPEAT
BB3C  CF A6 9C   08D4  A6 1C  RST 08  1CA6   KM GET REPEAT
BB3F  CF 6D 9C   08D6  6D 1C  RST 08  1C6D   KM SET DELAY
BB42  CF 69 9C   08D8  69 1C  RST 08  1C69   KM GET DELAY
BB45  CF 71 9C   08DA  71 1C  RST 08  1C71   KM ARM BREAK
BB48  CF 82 9C   08DC  82 1C  RST 08  1C82   KM DISARM BREAK
BB4B  CF 90 9C   08DE  90 1C  RST 08  1C90   KM BREAK EVENT
BB4E  CF 78 90   08E0  78 10  RST 08  1078   TXT INITIALIZE
BB51  CF 88 90   08E2  88 10  RST 08  1088   TXT RESET
BB54  CF 51 94   08E4  51 14  RST 08  1451   TXT VDU ENABLE
```

```
BB57  CF 4B 94   08E6  4B 14   RST 08  144B   TXT VDU DISABLE
BB5A  CF 00 94   08E8  00 14   RST 08  1400   TXT OUTPUT
BB5D  CF 34 93   08EA  34 13   RST 08  1334   TXT WR CHAR
BB60  CF AB 93   08EC  AB 13   RST 08  13AB   TXT RD CHAR
BB63  CF A7 93   08EE  A7 13   RST 08  13A7   TXT SET GRAPHIC
BB66  CF 0C 92   08F0  0C 12   RST 08  120C   TXT WIN ENABLE
BB69  CF 56 92   08F2  56 12   RST 08  1256   TXT GET WINDOW
BB6C  CF 40 95   08F4  40 15   RST 08  1540   TXT CLEAR WINDOW
BB6F  CF 5E 91   08F6  5E 11   RST 08  115E   TXT SET COLUMN
BB72  CF 69 91   08F8  69 11   RST 08  1169   TXT SET ROW
BB75  CF 74 91   08FA  74 11   RST 08  1174   TXT SET CURSOR
BB78  CF 80 91   08FC  80 11   RST 08  1180   TXT GET CURSOR
BB7B  CF 89 92   08FE  89 12   RST 08  1289   TXT CUR ENABLE
BB7E  CF 9A 92   0900  9A 12   RST 08  129A   TXT CUR DISABLE
BB81  CF 79 92   0902  79 12   RST 08  1279   TXT CUR ON
BB84  CF 81 92   0904  81 12   RST 08  1281   TXT CUR OFF
BB87  CF CE 91   0906  CE 11   RST 08  11CE   TXT VALIDATE
BB8A  CF 68 92   0908  68 12   RST 08  1268   TXT PLACE CURSOR
BB8D  CF 68 92   090A  68 12   RST 08  1268   TXT REMOVE CURSOR
BB90  CF A9 92   090C  A9 12   RST 08  12A9   TXT SET PEN
BB93  CF BD 92   090E  BD 12   RST 08  12BD   TXT GET PEN
BB96  CF AE 92   0910  AE 12   RST 08  12AE   TXT SET PAPER
BB99  CF C3 92   0912  C3 12   RST 08  12C3   TXT GET PAPER
BB9C  CF C9 92   0914  C9 12   RST 08  12C9   TXT INVERSE
BB9F  CF 7A 93   0916  7A 13   RST 08  137A   TXT SET BACK
BBA2  CF 87 93   0918  87 13   RST 08  1387   TXT GET BACK
BBA5  CF D3 92   091A  D3 12   RST 08  12D3   TXT GET MATRIX
BBA8  CF F1 92   091C  F1 12   RST 08  12F1   TXT SET MATRIX
BBAB  CF FD 92   091E  FD 12   RST 08  12FD   TXT SET M TABLE
BBAE  CF 2A 93   0920  2A 13   RST 08  132A   TXT GET M TABLE
BBB1  CF CB 94   0922  CB 14   RST 08  14CB   TXT GET CONTROLS
BBB4  CF E8 90   0924  E8 10   RST 08  10E8   TXT STR SELECT
BBB7  CF 07 91   0926  07 11   RST 08  1107   TXT SWAP STREAMS
BBBA  CF B0 95   0928  B0 15   RST 08  15B0   GRA INITIALIZE
BBBD  CF DF 95   092A  DF 15   RST 08  15DF   GRA RESET
BBC0  CF F4 95   092C  F4 15   RST 08  15F4   GRA MOVE ABSOLUTE
BBC3  CF F1 95   092E  F1 15   RST 08  15F1   GRA MOVE RELATIVE
BBC6  CF FC 95   0930  FC 15   RST 08  15FC   GRA ASK CURSOR
BBC9  CF 04 96   0932  04 16   RST 08  1604   GRA SET ORIGIN
BBCC  CF 12 96   0934  12 16   RST 08  1612   GRA GET ORIGIN
BBCF  CF 34 97   0936  34 17   RST 08  1734   GRA WIN WIDTH
BBD2  CF 79 97   0938  79 17   RST 08  1779   GRA WIN HEIGHT
BBD5  CF A6 97   093A  A6 17   RST 08  17A6   GRA GET WINDOW WIDTH
BBD8  CF BC 97   093C  BC 17   RST 08  17BC   GRA GET WINDOW HEIGHT
BBDB  CF C5 97   093E  C5 17   RST 08  17C5   GRA CLEAR WINDOW
BBDE  CF F6 97   0940  F6 17   RST 08  17F6   GRA SET PEN
BBE1  CF 04 98   0942  04 18   RST 08  1804   GRA GET PEN
BBE4  CF FD 97   0944  FD 17   RST 08  17FD   GRA SET PAPER
BBE7  CF 0A 98   0946  0A 18   RST 08  180A   GRA GET PAPER
BBEA  CF 13 98   0948  13 18   RST 08  1813   GRA PLOT ABSOLUTE
BBED  CF 10 98   094A  10 18   RST 08  1810   GRA PLOT RELATIVE
BBF0  CF 27 98   094C  27 18   RST 08  1827   GRA TEST ABSOLUTE
BBF3  CF 24 98   094E  24 18   RST 08  1824   GRA TEST RELATIVE
BBF6  CF 39 98   0950  39 18   RST 08  1839   GRA LINE ABSOLUTE
BBF9  CF 36 98   0952  36 18   RST 08  1836   GRA LINE RELATIVE
BBFC  CF 45 99   0954  45 19   RST 08  1945   GRA WR CHAR
BBFF  CF A0 8A   0956  A0 0A   RST 08  0AA0   SCR INITIALIZE
BC02  CF B1 8A   0958  B1 0A   RST 08  0AB1   SCR RESET
```

```
BC05  CF 3C 8B  095A  3C 0B  RST 08  0B3C  SCR SET OFFSET
BC08  CF 45 8B  095C  45 0B  RST 08  0B45  SCR SET BASE
BC0B  CF 50 8B  095E  50 0B  RST 08  0B50  SCR GET LOCATION
BC0E  CF CA 8A  0960  CA 0A  RST 08  0ACA  SCR SET MODE
BC11  CF EC 8A  0962  EC 0A  RST 08  0AEC  SCR GET MODE
BC14  CF F7 8A  0964  F7 0A  RST 08  0AF7  SCR MODE CLEAR
BC17  CF 57 8B  0966  57 0B  RST 08  0B57  SCR CHAR LIMITS
BC1A  CF 64 8B  0968  64 0B  RST 08  0B64  SCR CHAR POSITION
BC1D  CF A9 8B  096A  A9 0B  RST 08  0BA9  SCR DOT POSITION
BC20  CF F9 8B  096C  F9 0B  RST 08  0BF9  SCR NEXT BYTE
BC23  CF 05 8C  096E  05 0C  RST 08  0C05  SCR PREV BYTE
BC26  CF 13 8C  0970  13 0C  RST 08  0C13  SCR NEXT LINE
BC29  CF 2D 8C  0972  2D 0C  RST 08  0C2D  SCR PREV LINE
BC2C  CF 86 8C  0974  86 0C  RST 08  0C86  SCR INK ENCODE
BC2F  CF A0 8C  0976  A0 0C  RST 08  0CA0  SCR INK DECODE
BC32  CF EC 8C  0978  EC 0C  RST 08  0CEC  SCR SET INK
BC35  CF 14 8D  097A  14 0D  RST 08  0D14  SCR GET INK
BC38  CF F1 8C  097C  F1 0C  RST 08  0CF1  SCR SET BORDER
BC3B  CF 19 8D  097E  19 0D  RST 08  0D19  SCR GET BORDER
BC3E  CF E4 8C  0980  E4 0C  RST 08  0CE4  SCR SET FLASHING
BC41  CF E8 8C  0982  E8 0C  RST 08  0CE8  SCR GET FLASHING
BC44  CF B3 8D  0984  B3 0D  RST 08  0DB3  SCR FILL BOX
BC47  CF B7 8D  0986  B7 0D  RST 08  0DB7  SCR FLOOD BOX
BC4A  CF DF 8D  0988  DF 0D  RST 08  0DDF  SCR CHAR INVERT
BC4D  CF FA 8D  098A  FA 0D  RST 08  0DFA  SCR HARDWARE ROLL
BC50  CF 3E 8E  098C  3E 0E  RST 08  0E3E  SCR SOFTWARE ROLL
BC53  CF F3 8E  098E  F3 0E  RST 08  0EF3  SCR UNPACK
BC56  CF 49 8F  0990  49 0F  RST 08  0F49  SCR REPACK
BC59  CF 49 8C  0992  49 0C  RST 08  0C49  SCR ACCESS
BC5C  CF 6B 8C  0994  6B 0C  RST 08  0C6B  SCR PIXELS
BC5F  CF C4 8F  0996  C4 0F  RST 08  0FC4  SCR HORIZONTAL
BC62  CF 2F 90  0998  2F 10  RST 08  102F  SCR VERTICAL
BC65  CF 70 A3  099A  70 23  RST 08  2370  CAS INITIALIZE
BC68  CF 7F A3  099C  7F 23  RST 08  237F  CAS SET SPEED
BC6B  CF 8E A3  099E  8E 23  RST 08  238E  CAS NOISY
BC6E  CF 4B AA  09A0  4B 2A  RST 08  2A4B  CAS START MOTOR
BC71  CF 4F AA  09A2  4F 2A  RST 08  2A4F  CAS STOP MOTOR
BC74  CF 51 AA  09A4  51 2A  RST 08  2A51  CAS RESTORE MOTOR
BC77  CF 92 A3  09A6  92 23  RST 08  2392  CAS IN OPEN
BC7A  CF FC A3  09A8  FC 23  RST 08  23FC  CAS IN CLOSE
BC7D  CF 01 A4  09AA  01 24  RST 08  2401  CAS IN ABANDON
BC80  CF 35 A4  09AC  35 24  RST 08  2435  CAS IN CHAR
BC83  CF AB A4  09AE  AB 24  RST 08  24AB  CAS IN DIRECT
BC86  CF 9A A4  09B0  9A 24  RST 08  249A  CAS RETURN
BC89  CF 96 A4  09B2  96 24  RST 08  2496  CAS TEST EOF
BC8C  CF AB A3  09B4  AB 23  RST 08  23AB  CAS OUT OPEN
BC8F  CF 15 A4  09B6  15 24  RST 08  2415  CAS OUT CLOSE
BC92  CF 2E A4  09B8  2E 24  RST 08  242E  CAS OUT ABANDON
BC95  CF 5B A4  09BA  5B 24  RST 08  245B  CAS OUT CHAR
BC98  CF EA A4  09BC  EA 24  RST 08  24EA  CAS OUT DIRECT
BC9B  CF 28 A5  09BE  28 25  RST 08  2528  CAS CATALOG
BC9E  CF 3F A8  09C0  3F 28  RST 08  283F  CAS WRITE
BCA1  CF 36 A8  09C2  36 28  RST 08  2836  CAS READ
BCA4  CF 51 A8  09C4  51 28  RST 08  2851  CAS CHECK
BCA7  CF 68 9E  09C6  68 1E  RST 08  1E68  SOUND RESET
BCAA  CF 9F 9F  09C8  9F 1F  RST 08  1F9F  SOUND QUEUE
BCAD  CF 6C A0  09CA  6C 20  RST 08  206C  SOUND CHECK
BCB0  CF 89 A0  09CC  89 20  RST 08  2089  SOUND ARM EVENT
```

```
BCB3  CF 4A A0  09CE  4A 20  RST 08  204A  SOUND RELEASE
BCB6  CF CB 9E  09D0  CB 1E  RST 08  1ECB  SOUND HOLD
BCB9  CF E6 9E  09D2  E6 1E  RST 08  1EE6  SOUND CONTINUE
BCBC  CF 38 A3  09D4  38 23  RST 08  2338  SOUND AMPL ENVELOPE
BCBF  CF 3D A3  09D6  3D 23  RST 08  233D  SOUND TONE ENVELOPE
BCC2  CF 49 A3  09D8  49 23  RST 08  2349  SOUND A ADDRESS
BCC5  CF 4E A3  09DA  4E 23  RST 08  234E  SOUND T ADDRESS
BCC8  CF 5C 80  09DC  5C 00  RST 08  005C  KL CHOKE OFF
BCCB  CF 29 83  09DE  29 03  RST 08  0329  KL ROM WALK
BCCE  CF 32 83  09E0  32 03  RST 08  0332  KL INIT BACK
BCD1  CF A1 82  09E2  A1 02  RST 08  02A1  KL LOG EXT
BCD4  CF B2 82  09E4  B2 02  RST 08  02B2  KL FIND COMMAND
BCD7  CF 63 81  09E6  63 01  RST 08  0163  KL NEW FRAME FLY
BCDA  CF 6A 81  09E8  6A 01  RST 08  016A  KL ADD FRAME FLY
BCDD  CF 70 81  09EA  70 01  RST 08  0170  KL DELETE FRAME FLY
BCE0  CF 76 81  09EC  76 01  RST 08  0176  KL NEW FAST TICKER
BCE3  CF 7D 81  09EE  7D 01  RST 08  017D  KL ADD FAST TICKER
BCE6  CF 83 81  09F0  83 01  RST 08  0183  KL DELETE FAST TICKER
BCE9  CF B3 81  09F2  B3 01  RST 08  01B3  KL ADD TICKER
BCEC  CF C5 81  09F4  C5 01  RST 08  01C5  KL DELETE TICKER
BCEF  CF D2 81  09F6  D2 01  RST 08  01D2  KL INIT EVENT
BCF2  CF E2 81  09F8  E2 01  RST 08  01E2  KL EVENT
BCF5  CF 28 82  09FA  28 02  RST 08  0228  KL SYNC RESET
BCF8  CF 85 82  09FC  85 02  RST 08  0285  KL DEL SYNCHRONOUS
BCFB  CF 56 82  09FE  56 02  RST 08  0256  KL NEXT SYNC
BCFE  CF 1A 82  0A00  1A 02  RST 08  021A  KL DO SYNC
BD01  CF 77 82  0A02  77 02  RST 08  0277  KL DONE SYNC
BD04  CF 95 82  0A04  95 02  RST 08  0295  KL EVENT DISABLE
BD07  CF 9B 82  0A06  9B 02  RST 08  029B  KL EVENT ENABLE
BD0A  CF 8E 82  0A08  8E 02  RST 08  028E  KL DISARM EVENT
BD0D  CF 99 80  0A0A  99 00  RST 08  0099  KL TIME PLEASE
BD10  CF A3 80  0A0C  A3 00  RST 08  00A3  KL TIME SET
BD13  CF DC 85  0A0E  DC 05  RST 08  05DC  MC BOOT PROGRAM
BD16  CF 0B 86  0A10  0B 06  RST 08  060B  MC START PROGRAM
BD19  CF BA 87  0A12  BA 07  RST 08  07BA  MC WAIT FLYBACK
BD1C  CF 76 87  0A14  76 07  RST 08  0776  MC SET MODE
BD1F  CF C6 87  0A16  C6 07  RST 08  07C6  MC SCREEN OFFSET
BD22  CF 86 87  0A18  86 07  RST 08  0786  MC CLEAR INKS
BD25  CF 99 87  0A1A  99 07  RST 08  0799  MC SET INKS
BD28  CF E6 87  0A1C  E6 07  RST 08  07E6  MC RESET PRINTER
BD2B  CF F2 87  0A1E  F2 07  RST 08  07F2  MC PRINT CHAR
BD2E  CF 1B 88  0A20  1B 08  RST 08  081B  MC BUSY PRINTER
BD31  CF 07 88  0A22  07 08  RST 08  0807  MC SEND PRINTER
BD34  CF 26 88  0A24  26 08  RST 08  0826  MC SOUND REGISTER
BD37  CF 88 88  0A26  88 08  RST 08  0888  JUMP RESTORE
BD3A  CF 98 AA  0A28  98 2A  RST 08  2A98  EDIT
BD3D  EF 18 2E  0A2A  18 2E  RST 28  2E18  FLO Zahl kopieren
BD40  EF 29 2E  0A2C  29 2E  RST 28  2E29  FLO INT nach REAL
BD43  EF 55 2E  0A2E  55 2E  RST 28  2E55  FLO 4-Bytes nach REAL
BD46  EF 66 2E  0A30  66 2E  RST 28  2E66  FLO REAL nach INTEGER
BD49  EF 8E 2E  0A32  8E 2E  RST 28  2E8E  FLO Zahl runden
BD4C  EF A1 2E  0A34  A1 2E  RST 28  2EA1  FLO Nachkommast. abschn.
BD4F  EF AC 2E  0A36  AC 2E  RST 28  2EAC  FLO INT-Funktion
BD52  EF B6 2E  0A38  B6 2E  RST 28  2EB6  FLO Params f. Dez.-Wand.
BD55  EF 1D 2F  0A3A  1D 2F  RST 28  2F1D  FLO Zahl mit 10^A mult.
BD58  EF 3F 33  0A3C  3F 33  RST 28  333F  FLO (HL):=(HL)+(DE)
BD5B  EF 37 33  0A3E  37 33  RST 28  3337  FLO (HL):=(HL)-(DE)
BD5E  EF 3B 33  0A40  3B 33  RST 28  333B  FLO (HL):=(DE)-(HL)
```

```
BD61  EF 15 34   0A42  15 34   RST 28  3415   FLO (HL):=(HL)*(DE)
BD64  EF 9E 34   0A44  9E 34   RST 28  349E   FLO (HL):=(HL)/(DE)
BD67  EF 78 35   0A46  78 35   RST 28  3578   FLO (HL):=(HL)*2^A
BD6A  EF 9A 35   0A48  9A 35   RST 28  359A   FLO Vergleich (HL)-(DE)
BD6D  EF F8 35   0A4A  F8 35   RST 28  35F8   Vorzeichen invertieren
BD70  EF E8 35   0A4C  E8 35   RST 28  35E8   FLO SGN-Funktion
BD73  EF AE 31   0A4E  AE 31   RST 28  31AE   FLO DEG/RAD
BD76  EF A3 31   0A50  A3 31   RST 28  31A3   FLO PI-Funktion
BD79  EF 0A 31   0A52  0A 31   RST 28  310A   FLO SQR-Funktion
BD7C  EF 0D 31   0A54  0D 31   RST 28  310D   FLO Potenzierung
BD7F  EF 14 30   0A56  14 30   RST 28  3014   FLO LOG-Funktion
BD82  EF 0F 30   0A58  0F 30   RST 28  300F   FLO LOG10-Funktion
BD85  EF 90 30   0A5A  90 30   RST 28  3090   FLO EXP-Funktion
BD88  EF BC 31   0A5C  BC 31   RST 28  31BC   FLO SIN-Funktion
BD8B  EF B2 31   0A5E  B2 31   RST 28  31B2   FLO COS-Funktion
BD8E  EF 31 32   0A60  31 32   RST 28  3231   FLO TAN-Funktion
BD91  EF 41 32   0A62  41 32   RST 28  3241   FLO ATN-Funktion
BD94  EF 5E 2E   0A64  5E 2E   RST 28  2E5E   FLO 5 Bytes nach REAL
BD97  EF 94 2F   0A66  94 2F   RST 28  2F94   FLO RND INITIALIZE
BD9A  EF A1 2F   0A68  A1 2F   RST 28  2FA1   FLO RND SEED
BD9D  EF B7 2F   0A6A  B7 2F   RST 28  2FB7   FLO RND-Funktion
BDA0  EF E6 2F   0A6C  E6 2F   RST 28  2FE6   FLO letzter RND-Wert
BDA3  EF 08 37   0A6E  08 37   RST 28  3708   INT Params f. Dez.-Wand.
BDA6  EF 0E 37   0A70  0E 37   RST 28  370E   INT Dez.-Par. f. pos. Z.
BDA9  EF 15 37   0A72  15 37   RST 28  3715   INT sig. Bin.>2er Komp.
BDAC  EF 28 37   0A74  28 37   RST 28  3728   INT HL:=HL+DE
BDAF  EF 31 37   0A76  31 37   RST 28  3731   INT HL:=HL-DE
BDB2  EF 30 37   0A78  30 37   RST 28  3730   INT HL:=DE-HL
BDB5  EF 39 37   0A7A  39 37   RST 28  3739   INT HL:=HL*DE
BDB8  EF 7A 37   0A7C  7A 37   RST 28  377A   INT HL:=HL DIV DE
BDBB  EF 81 37   0A7E  81 37   RST 28  3781   INT HL:=HL MOD DE
BDBE  EF 50 37   0A80  50 37   RST 28  3750   vorzeichenlose Multipl.
BDC1  EF 8C 37   0A82  8C 37   RST 28  378C   vorzeichenlose Division
BDC4  EF E9 37   0A84  E9 37   RST 28  37E9   INT Vergleich HL-DE
BDC7  EF D4 37   0A86  D4 37   RST 28  37D4   INT HL:=-HL
BDCA  EF E0 37   0A88  E0 37   RST 28  37E0   INT A:=SGN(HL)

***********************************     Indirection(s) kopieren
                                        IN : HL: Zeiger auf Tabelle
                                             (HL): Zahl d. zu kop. Bytes
                                             (HL+1): Zieladresse im RAM
                                             ab (HL+3): Ansprünge f. Ind.
0A8A  4E              LD     C,(HL)       Zahl der Bytes
0A8B  06 00           LD     B,00         Zahl der Bytes hi =0
0A8D  23              INC    HL
0A8E  5E              LD     E,(HL)         Zieladresse im RAM
0A8F  23              INC    HL             nach DE
0A90  56              LD     D,(HL)
0A91  23              INC    HL
0A92  ED B0           LDIR                Indirection(s) kopieren
0A94  C9              RET

0A95  C7              RST    00
0A96  C7              RST    00
0A97  C7              RST    00
0A98  C7              RST    00
0A99  C7              RST    00
0A9A  C7              RST    00
```

```
0A9B  C7          RST    00
0A9C  C7          RST    00
0A9D  C7          RST    00
0A9E  C7          RST    00
0A9F  C7          RST    00


------------------------ SCREEN PACK (SCR) ------------------------

**********************************   SCR INITIALIZE
0AA0  11 4D 10    LD     DE,104D        Zeiger auf Default-Farbwerte
0AA3  CD 86 07    CALL   0786           Farbstifte und Rand setzen
0AA6  3E C0       LD     A,C0           Bildschirmspeicher ab $C000
0AA8  32 CB B1    LD     (B1CB),A       SCR BASE setzen
0AAB  CD B1 0A    CALL   0AB1           SCR RESET, Tabellen initialisieren
0AAE  C3 F2 0A    JP     0AF2           Mode 1 einschalten

**********************************   SCR RESET
                                     (Tabellen initialisieren)
0AB1  AF          XOR    A              Null
0AB2  CD 49 0C    CALL   0C49           Force-Mode f. SCR WRITE
0AB5  21 BE 0A    LD     HL,0ABE        Adresse der ROM-Tabelle
0AB8  CD 8A 0A    CALL   0A8A           Indirections kopieren
0ABB  C3 D2 0C    JP     0CD2           Farbwerttabellen initialisieren

0ABE  09                                Anzahl der zu kopierenden Bytes
0ABF  E5 BD                             Zieladresse im RAM
0AC1  C3 82 0C    JP     0C82           SCR READ
0AC4  C3 68 0C    JP     0C68           SCR WRITE
0AC7  C3 F7 0A    JP     0AF7           SCR MODE CLEAR

**********************************   SCR SET MODE
                                     (Mode einstellen)
                                     IN : A: Mode-Nummer
0ACA  E6 03       AND    03             Mode von 0 bis 3
0ACC  FE 03       CP     03             Mode >=3 ?
0ACE  D0          RET    NC             ja? dann zurück
0ACF  F5          PUSH   AF             Mode retten
0AD0  CD 4F 0D    CALL   0D4F           Event-Block f. Farbwechsel aushängen
0AD3  F1          POP    AF             Mode
0AD4  5F          LD     E,A            nach E
0AD5  CD B7 10    CALL   10B7           bei allen Windows Paper/Pen decod.
0AD8  F5          PUSH   AF             aktuelle Windownummer
0AD9  CD D6 15    CALL   15D6           Paper u. Pen der Graphik holen
0ADC  E5          PUSH   HL             und retten
0ADD  7B          LD     A,E            Mode
0ADE  CD 11 0B    CALL   0B11           Bitmasken laden, Mode einschalten
0AE1  CD EB BD    CALL   BDEB           Bildschirm löschen, Event-Block einh.
0AE4  E1          POP    HL             Pen/Paper von Graphik
0AE5  CD B6 15    CALL   15B6           wieder setzen
0AE8  F1          POP    AF             aktuelle Windownummer
0AE9  C3 D5 10    JP     10D5           Farben codieren, Windows auf Default
```

```
***********************************    SCR GET MODE
                                       (Mode-Nummer holen)
                                       OUT: A: Mode-Nummer
                                            CY=1 u. Z=0 f. Mode 0
                                            CY=0 u. Z=1 f. Mode 1
                                            CY=0 u. Z=0 f. Mode 2
0AEC  3A C8 B1    LD     A,(B1C8)         Mode-Nummer laden
0AEF  FE 01       CP     01               Flags entsprechend setzen
0AF1  C9          RET

***********************************    Mode 1 einschalten
0AF2  3E 01       LD     A,01             Mode-Nummer
0AF4  CD 11 0B    CALL   0B11             Bitmasken laden, Mode einschalten

***********************************    SCR MODE CLEAR
                                       (Bildschirm löschen, Event-Block einh.)
0AF7  CD 4F 0D    CALL   0D4F             Event-Block f. Farbwechsel aush.
0AFA  21 00 00    LD     HL,0000          SCR OFFSET=0
0AFD  CD 3C 0B    CALL   0B3C             SCR OFFSET setzen
0B00  2A CA B1    LD     HL,(B1CA)        SCR BASE
0B03  2E 00       LD     L,00             Lo-Byte löschen
0B05  54          LD     D,H                Adresse +1
0B06  1E 01       LD     E,01               nach DE
0B08  01 FF 3F    LD     BC,3FFF          Zähler f. 16K-Bereich
0B0B  75          LD     (HL),L           1. Byte löschen
0B0C  ED B0       LDIR                    16K-Block löschen (Farbstift 0)
0B0E  C3 3C 0D    JP     0D3C             Farbwechsel-Event generieren u. einh.

***********************************    Bitmasken laden, Mode einschalten
                                       IN : A: Mode-Nummer
0B11  21 3A 0B    LD     HL,0B3A          Adresse f. Mode 0
0B14  FE 01       CP     01
0B16  38 08       JR     C,0B20           Mode 0?
0B18  21 36 0B    LD     HL,0B36          Adresse f. Mode 1
0B1B  28 03       JR     Z,0B20           Mode 1?
0B1D  21 2E 0B    LD     HL,0B2E          Adresse f. Mode 2
0B20  11 CF B1    LD     DE,B1CF          Zeiger auf Bitmasken im RAM
0B23  01 08 00    LD     BC,0008          Zahl der Bytes
0B26  ED B0       LDIR                    Bitmasken ins RAM kopieren
0B28  32 C8 B1    LD     (B1C8),A         Mode speichern
0B2B  C3 76 07    JP     0776             und an Gate Array übergeben

0B2E  80 40 20 10 08 04 02 01             Bitmasken f. Mode 2
0B36  88 44 22 11                         Bitmasken f. Mode 1
0B3A  AA 55                               Bitmasken f. Mode 0

***********************************    SCR SET OFFSET
                                       (SCR OFFSET setzen)
                                       IN : HL: SCR OFFSET
0B3C  7C          LD     A,H                RA-Bits löschen,
0B3D  E6 07       AND    07                 oberste Rasterzeile
0B3F  67          LD     H,A                des Zeichenblocks auswählen
0B40  22 C9 B1    LD     (B1C9),HL        SCR OFFSET speichern
0B43  18 05       JR     0B4A             und an CRTC übergeben

***********************************    SCR SET BASE
                                       (SCR BASE setzen)
                                       IN : A: SCR BASE (hi)
```

```
0B45  E6 C0         AND    C0          oberste Bits isolieren
0B47  32 CB B1      LD     (B1CB),A    SCR BASE speichern
0B4A  CD 50 0B      CALL   0B50        SCR OFFSET holen
0B4D  C3 C6 07      JP     07C6        Adressen an CRTC übergeben

**********************************   SCR GET LOCATION
                                     OUT: HL: SCR OFFSET
                                           A: SCR BASE
0B50  2A C9 B1      LD     HL,(B1C9)   SCR OFFSET laden
0B53  3A CB B1      LD     A,(B1CB)    SCR BASE laden
0B56  C9            RET

**********************************   SCR CHAR LIMITS
                                     OUT: B: max. Spalte
                                          C: max. Zeile
0B57  CD EC 0A      CALL   0AEC        Mode holen
0B5A  01 18 13      LD     BC,1318     Spalte 19, Zeile 24
0B5D  D8            RET    C           Mode 0?
0B5E  06 27         LD     B,27        Spalte 39
0B60  C8            RET    Z           Mode 1?
0B61  06 4F         LD     B,4F        Spalte 79
0B63  C9            RET

**********************************   SCR CHAR POSITION
                                     IN : H: Spalte
                                          L: Zeile
                                     OUT: HL: Adresse der Zeichenpos.
                                          B: Anz. d. Bytes pro Zeichen
0B64  D5            PUSH   DE
0B65  CD EC 0A      CALL   0AEC        Mode holen
0B68  06 04         LD     B,04        4 Bytes pro Zeichen
0B6A  38 05         JR     C,0B71      Mode 0?
0B6C  06 02         LD     B,02        2 Bytes pro Zeichen
0B6E  28 01         JR     Z,0B71      Mode 1?
0B70  05            DEC    B           1 Byte pro Zeichen
0B71  C5            PUSH   BC          Zahl d. Bytes / Zeichen retten
0B72  5C            LD     E,H         Spalte
0B73  16 00         LD     D,00        Spalte hi=0 setzen
0B75  62            LD     H,D         Zeile hi=0 setzen
0B76  D5            PUSH   DE          Spalte retten
0B77  54            LD     D,H           Zeile
0B78  5D            LD     E,L           nach DE
0B79  29            ADD    HL,HL
0B7A  29            ADD    HL,HL         Zeile
0B7B  19            ADD    HL,DE         mit 80
0B7C  29            ADD    HL,HL         (80 Bytes pro Zeile)
0B7D  29            ADD    HL,HL         multiplizieren
0B7E  29            ADD    HL,HL
0B7F  29            ADD    HL,HL
0B80  D1            POP    DE          Spalte
0B81  19            ADD    HL,DE         Spalte je nach Mode
0B82  10 FD         DJNZ   0B81          1-, 2- oder 4-fach addieren
0B84  ED 5B C9 B1   LD     DE,(B1C9)   SCR OFFSET
0B88  19            ADD    HL,DE       addieren
0B89  7C            LD     A,H           Übertrag
0B8A  E6 07         AND    07            auf RA-Bits
0B8C  67            LD     H,A           löschen
0B8D  3A CB B1      LD     A,(B1CB)    SCR BASE
```

```
0B90  84            ADD    H           zu Hi-Byte addieren
0B91  67            LD     H,A         =Adresse der Zeichenposition
0B92  C1            POP    BC          Zahl der Bytes pro Zeichen
0B93  D1            POP    DE
0B94  C9            RET

**********************************  Window-Parameter berechnen
                                    IN : H: linke Grenze
                                         L: obere Grenze
                                         D: rechte Grenze
                                         E: untere Grenze
                                    OUT: HL: Bildschirmadr. links ob.
                                         D: Zahl der Bytes pro Zeile
                                         E: Rasterzeilenzahl im Wind.
0B95  7B            LD     A,E         untere Grenze
0B96  95            SUB    L           - obere Grenze
0B97  3C            INC    A           +1 =Zahl der Zeichenzeilen
0B98  87            ADD    A             mal 8
0B99  87            ADD    A             ergibt
0B9A  87            ADD    A             Zahl der Rasterzeilen
0B9B  5F            LD     E,A         nach E
0B9C  7A            LD     A,D         rechte Grenze
0B9D  94            SUB    H           - linke Grenze
0B9E  3C            INC    A           +1 = Zahl der Zeichen / Zeile
0B9F  57            LD     D,A         nach D
0BA0  CD 64 0B      CALL   0B64        Zahl der Bytes / Zeichen n. B
0BA3  AF            XOR    A             Zahl der Zeichen pro Zeile
0BA4  82            ADD    D             mit Zahl der Bytes
0BA5  10 FD         DJNZ   0BA4          pro Zeichen multiplizieren
0BA7  57            LD     D,A         nach D
0BA8  C9            RET

**********************************  SCR DOT POSITION
                                    IN : HL: Y-Koordinate
                                         DE: X-Koordinate
                                    OUT: HL: Bildschirmadresse
                                         C: Maske für diesen Punkt
                                         DE: Adresse der Maske
                                         B: Maske für signif. Bits f.
                                            Punktstell. innerh. Byte
0BA9  D5            PUSH   DE          X-Koordinate retten
0BAA  EB            EX     DE,HL
0BAB  21 C7 00      LD     HL,00C7     maximale Y-Koordinate
0BAE  B7            OR     A           Carry löschen
0BAF  ED 52         SBC    HL,DE       Y-Koordinate wandeln, oben =0
0BB1  7D            LD     A,L         Y-Koordinate lo
0BB2  E6 07         AND    07          RA-Bits isolieren
0BB4  87            ADD    A             von Bit 0 bis Bit 2
0BB5  87            ADD    A             nach Bit 3 bis Bit 5
0BB6  87            ADD    A             (RA-Bit-Stellung für CRTC)
0BB7  4F            LD     C,A         RA-Bits nach C
0BB8  7D            LD     A,L         Y-Koordinate lo
0BB9  E6 F8         AND    F8            RA-Bits
0BBB  6F            LD     L,A           löschen
0BBC  54            LD     D,H         Y-Koordinate hi
0BBD  5D            LD     E,L         und lo nach DE
0BBE  29            ADD    HL,HL         mal 10, da 80 Bytes
0BBF  29            ADD    HL,HL         pro Zeichenzeile
```

```
0BC0  19           ADD   HL,DE         und 8 Graphik- (Raster-)
0BC1  29           ADD   HL,HL         Zeilen pro Zeichenzeile
0BC2  D1           POP   DE          X-Koordinate
0BC3  CD EC 0A     CALL  0AEC        Mode holen
0BC6  06 01        LD    B,01        Maske für Bit 0
0BC8  38 06        JR    C,0BD0      Mode 0?
0BCA  06 03        LD    B,03        Maske für Bit 0 und Bit 1
0BCC  28 02        JR    Z,0BD0      Mode 1?
0BCE  06 07        LD    B,07        Maske für Bit 0 bis Bit 2
0BD0  78           LD    A,B         Maske f. Punktstellung im Byte
0BD1  A3           AND   E           Punktstell. innerh. Byte isol.
0BD2  F5           PUSH  AF          und retten
0BD3  78           LD    A,B         Maske f. Punktstellung im Byte
0BD4  0F           RRCA
0BD5  CB 3A        SRL   D             X-Koordinate
0BD7  CB 1B        RR    E             halbieren
0BD9  0F           RRCA              weit. Bits f. Punktst. sign.?
0BDA  38 F9        JR    C,0BD5      dann diese eliminieren
0BDC  19           ADD   HL,DE       zu Adresse der Zeile addieren
0BDD  ED 5B C9 B1  LD    DE,(B1C9)   SCR OFFSET
0BE1  19           ADD   HL,DE       addieren
0BE2  7C           LD    A,H          Übertrag
0BE3  E6 07        AND   07           auf RA-Bits
0BE5  67           LD    H,A          löschen
0BE6  3A CB B1     LD    A,(B1CB)    SCR BASE
0BE9  84           ADD   H           zu Hi-Byte addieren
0BEA  81           ADD   C           RA-Bits addieren
0BEB  67           LD    H,A         ergibt Bildschirmadresse
0BEC  F1           POP   AF          Punktstellung innerhalb Byte
0BED  E5           PUSH  HL          Bildschirmadresse retten
0BEE  16 00        LD    D,00        Punktstellung hi löschen
0BF0  5F           LD    E,A         Punktstellung lo
0BF1  21 CF B1     LD    HL,B1CF     Adresse der Tab. der Bitmasken
0BF4  19           ADD   HL,DE       Punktstellung (Maskenr.) add.
0BF5  4E           LD    C,(HL)      Maske laden
0BF6  EB           EX    DE,HL       Adresse der Maske nach DE
0BF7  E1           POP   HL          Bildschirmadresse nach HL
0BF8  C9           RET

************************************ SCR NEXT BYTE
                                   IN/OUT: HL: Bildschirmadresse
0BF9  2C           INC   L             Lo-Byte erhöhen
0BFA  C0           RET   NZ            kein Übertrag?
0BFB  24           INC   H             Hi-Byte erhöhen
0BFC  7C           LD    A,H
0BFD  E6 07        AND   07
0BFF  C0           RET   NZ            kein Übertrag auf RA-Bits?
0C00  7C           LD    A,H
0C01  D6 08        SUB   08            sonst RA-Bits wieder herstell.
0C03  67           LD    H,A
0C04  C9           RET

************************************ SCR PREV BYTE
                                   IN/OUT: HL: Bildschirmadresse
0C05  7D           LD    A,L           Lo-Byte
0C06  2D           DEC   L             erniedrigen
0C07  B7           OR    A
0C08  C0           RET   NZ            kein Übertrag?
```

```
0C09  7C        LD    A,H
0C0A  25        DEC   H            Hi-Byte erniedrigen
0C0B  E6 07     AND   07
0C0D  C0        RET   NZ           kein Übertrag auf RA-Bits?
0C0E  7C        LD    A,H
0C0F  C6 08     ADD   08           sonst RA-Bits wieder herstell.
0C11  67        LD    H,A
0C12  C9        RET

**********************************  SCR NEXT LINE
                                    IN/OUT: HL: Bildschirmadresse
0C13  7C        LD    A,H
0C14  C6 08     ADD   08           RA-Bits erhöhen
0C16  67        LD    H,A
0C17  E6 38     AND   38
0C19  C0        RET   NZ           kein Übertrag auf SCR BASE ?
0C1A  7C        LD    A,H
0C1B  D6 40     SUB   40           sonst SCR BASE wieder herst.
0C1D  67        LD    H,A
0C1E  7D        LD    A,L
0C1F  C6 50     ADD   50             Zeiger auf nächste Zeile
0C21  6F        LD    L,A            (80 Bytes/Z.)
0C22  D0        RET   NC           kein Übertrag?
0C23  24        INC   H            sonst Hi-Byte erhöhen
0C24  7C        LD    A,H
0C25  E6 07     AND   07
0C27  C0        RET   NZ           kein Übertrag auf RA-Bits?
0C28  7C        LD    A,H
0C29  D6 08     SUB   08           sonst RA-Bits wieder herstell.
0C2B  67        LD    H,A
0C2C  C9        RET

**********************************  SCR PREV LINE
                                    IN/OUT: HL: Bildschirmadresse
0C2D  7C        LD    A,H
0C2E  D6 08     SUB   08           RA-Bits erniedrigen
0C30  67        LD    H,A
0C31  E6 38     AND   38
0C33  FE 38     CP    38
0C35  C0        RET   NZ           kein Übertrag auf SCR BASE ?
0C36  7C        LD    A,H
0C37  C6 40     ADD   40           SCR BASE-Bits wieder herstell.
0C39  67        LD    H,A
0C3A  7D        LD    A,L
0C3B  D6 50     SUB   50             Zeiger auf vorige Zeile
0C3D  6F        LD    L,A            (80 Bytes/Z.)
0C3E  D0        RET   NC           kein Übertrag?
0C3F  7C        LD    A,H
0C40  25        DEC   H            sonst Hi-Byte erniedrigen
0C41  E6 07     AND   07
0C43  C0        RET   NZ           kein Übertrag auf RA-Bits?
0C44  7C        LD    A,H
0C45  C6 08     ADD   08           sonst RA-Bits wieder herstell.
0C47  67        LD    H,A
0C48  C9        RET
```

```
***********************************  SCR ACCESS
                                     IN : A: Nummer des Modus
0C49  E6 03      AND    03              Modus von 0 bis 3
0C4B  21 6B 0C   LD     HL,0C6B         Adresse für Force-Mode
0C4E  28 0F      JR     Z,0C5F          =0? dann Force-Mode
0C50  FE 02      CP     02
0C52  21 72 0C   LD     HL,0C72         Adresse für XOR-Mode
0C55  38 08      JR     C,0C5F          =1? dann XOR-Mode
0C57  21 77 0C   LD     HL,0C77         Adresse für AND-Mode
0C5A  28 03      JR     Z,0C5F          =2? dann AND-Mode
0C5C  21 7D 0C   LD     HL,0C7D         Adresse für OR-Mode
0C5F  3E C3      LD     A,C3            Opcode für JP
0C61  32 CC B1   LD     (B1CC),A        speichern
0C64  22 CD B1   LD     (B1CD),HL       Adresse für Indirection setzen
0C67  C9         RET

***********************************  SCR WRITE
                                     IN : HL: Adr. in Video-RAM
                                           B: Pen Byte
                                           C: Bitmaske für Pixelauswahl
0C68  C3 CC B1   JP     B1CC            Indirect. zu FORCE/XOR/AND/OR

***********************************  SCR PIXELS, Pixles forcieren
                                     IN : wie $0C68
0C6B  7E         LD     A,(HL)          Byte aus Video-RAM
0C6C  A8         XOR    B               Pen-Pixels in 0-Bits wandeln
0C6D  B1         OR     C               Bits aus Maske setzen
0C6E  A9         XOR    C               genau diese wieder löschen
0C6F  A8         XOR    B               0-Bits in Pen-Pixels wandeln
0C70  77         LD     (HL),A          und Byte wieder in Video-RAM
0C71  C9         RET

***********************************  XOR-Verknüpfung, Pixels invert.
                                     IN : wie $0C68
0C72  78         LD     A,B             Pen-Byte
0C73  A1         AND    C               Bits für Pixels isolieren
0C74  AE         XOR    (HL)            und diese invertieren
0C75  77         LD     (HL),A          Byte wieder in Video-RAM
0C76  C9         RET

***********************************  AND-Verknüpfung, Pixels löschen
                                     IN : wie $0C68
0C77  79         LD     A,C             Bitmaske
0C78  2F         CPL                    zu ändernde Bits auf 0 setzen
0C79  B0         OR     B               nur Pen-Pixels löschen
0C7A  A6         AND    (HL)            entsprechende Pixels löschen
0C7B  77         LD     (HL),A          Byte wieder in Video-RAM
0C7C  C9         RET

***********************************  OR-Verknüpfung, Pixels setzen
                                     IN : wie $0C68
0C7D  78         LD     A,B             Pen-Byte
0C7E  A1         AND    C               zu ändernde Bits isolieren
0C7F  B6         OR     (HL)            und diese setzen
0C80  77         LD     (HL),A          Byte wieder in Video-RAM
0C81  C9         RET
```

```
***********************************  SCR READ
                                     IN : HL: Bildschirmadresse
                                           C: Bitmaske für Pixel
                                     OUT: A: Farbstift-Nr.
0C82  7E          LD     A,(HL)         Byte aus Bildschirm
0C83  C3 AC 0C    JP     0CAC           Farbstift berechnen

***********************************  SCR INK ENCODE
                                     IN : A: Farbstift
                                     OUT: A: Farbmaske
0C86  C5          PUSH   BC
0C87  D5          PUSH   DE
0C88  CD C2 0C    CALL   0CC2           Bildsch.-Wert aus Farbst. ber.
0C8B  5F          LD     E,A            Bildschirm-Farbstift-Wert
0C8C  06 08       LD     B,08           Zähler für Maskenbits
0C8E  3A CF B1    LD     A,(B1CF)       1. Bitmaske für Pixelsauswahl
0C91  4F          LD     C,A            nach C
0C92  CB 0B       RRC    E              Bit aus Bildsch.-Farbstift
0C94  17          RLA                   in Farbmaske übertragen
0C95  CB 09       RRC    C              Bitmaske für Pixelauswahl
0C97  38 02       JR     C,0C9B         wieder Pixel 0 ? d. näch. Bit
0C99  CB 03       RLC    E              sonst dieses Bit f. nä. Pixel
0C9B  10 F5       DJNZ   0C92           weitere Bits ?
0C9D  D1          POP    DE
0C9E  C1          POP    BC
0C9F  C9          RET

***********************************  SCR INK DECODE
                                     IN : A: Farbmaske
                                     OUT: A: Farbstift
0CA0  C5          PUSH   BC
0CA1  47          LD     B,A            Farbmaske
0CA2  3A CF B1    LD     A,(B1CF)       1. Bitmaske für Pixelauswahl
0CA5  4F          LD     C,A            nach C
0CA6  78          LD     A,B            Farbmaske
0CA7  CD AC 0C    CALL   0CAC           Farbstift berechnen
0CAA  C1          POP    BC
0CAB  C9          RET

***********************************  Farbstift-Nummer berechnen
                                     IN : A: Farbmaske
                                          C: Bitmaske für Pixelauswahl
                                     OUT: A: Farbstift-Nummer
0CAC  D5          PUSH   DE
0CAD  11 08 00    LD     DE,0008        Zähler für 8 Farbmaskenbits
0CB0  0F          RRCA                  Bit aus Farbmaske
0CB1  CB 12       RL     D              in Bildsch.-Farbstift übertr.
0CB3  CB 09       RRC    C              Bitmaske für Pixelstellung
0CB5  38 02       JR     C,0CB9         Pixel 0 ?
0CB7  CB 1A       RR     D              sonst Maskenbit nicht übertr.
0CB9  1D          DEC    E
0CBA  20 F4       JR     NZ,0CB0        weitere Maskenbits ?
0CBC  7A          LD     A,D            Bildschirm-Farbstiftnr.
0CBD  CD C2 0C    CALL   0CC2           Farbstiftnummer berechnen
0CC0  D1          POP    DE
0CC1  C9          RET
```

```
**********************************   Farbstiftnummer konvertieren
                                     IN : A: Bildschirm-Wert
                                     OUT: A: Register-Wert
                                     bzw.
                                     IN : A: Register-Wert
                                     OUT: A: Bildschirm-Wert
0CC2  57          LD     D,A           Wert retten
0CC3  CD EC 0A    CALL   0AEC          Mode-Nummer holen
0CC6  7A          LD     A,D           Wert
0CC7  D0          RET    NC            nicht Mode 0 ?
0CC8  0F          RRCA
0CC9  0F          RRCA                   Bit 1 und Bit 2
0CCA  CE 00       ADC    00
0CCC  0F          RRCA
0CCD  9F          SBC    A               des Werts
0CCE  E6 06       AND    06
0CD0  AA          XOR    D               vertauschen
0CD1  C9          RET

**********************************   Farbtabellen initialisieren
0CD2  21 4D 10    LD     HL,104D       Zeiger auf Default-Werte
0CD5  11 D9 B1    LD     DE,B1D9       Zeiger auf RAM-Farbwerte
0CD8  01 22 00    LD     BC,0022       Länge für 2 Farbsätze
0CDB  ED B0       LDIR                 Werte kopieren
0CDD  AF          XOR    A               1. Farbsatz
0CDE  32 FB B1    LD     (B1FB),A        auswählen
0CE1  21 0A 0A    LD     HL,0A0A       Zähl-Werte gleichlang

**********************************   SCR SET FLASHING
                                     IN : L: Zählwert für 1. Farbsatz
                                          H: Zählwert für 2. Farbsatz
0CE4  22 D7 B1    LD     (B1D7),HL     Zählwerte setzen
0CE7  C9          RET

**********************************   SCR GET FLASHING
                                     OUT: L: Zählwert für 1. Farbsatz
                                          H: Zählwert für 2. Farbsatz
0CE8  2A D7 B1    LD     HL,(B1D7)     Zählwerte holen
0CEB  C9          RET

**********************************   SCR SET INK
                                     IN : A: Farbstiftnummer
                                          B: 1. Farbnummer
                                          C: 2. Farbnummer
0CEC  E6 0F       AND    0F              Farbstift von
0CEE  3C          INC    A               1 bis 16
0CEF  18 01       JR     0CF2          Farbnummern zuordnen

**********************************   SCR SET BORDER
                                     IN : B: 1. Farbnummer
                                          C: 2. Farbnummer
0CF1  AF          XOR    A             Farbstift 0 für BORDER
0CF2  5F          LD     E,A           Farbstiftnummer
0CF3  78          LD     A,B           1. Farbnummer
0CF4  CD 0A 0D    CALL   0D0A          Adresse in Farbmatrix holen
0CF7  46          LD     B,(HL)        Farbwert entsprechend Nr.
0CF8  79          LD     A,C           2. Farbwert
0CF9  CD 0A 0D    CALL   0D0A          Adresse in Farbmatrix holen
```

```
0CFC  4E         LD    C,(HL)       Farbwert entsprechend Nr.
0CFD  7B         LD    A,E          Farbstiftnummer
0CFE  CD 2F 0D   CALL  0D2F         Adressen in Tabellen holen
0D01  71         LD    (HL),C       2. Farbwert
0D02  EB         EX    DE,HL
0D03  70         LD    (HL),B       und 1. Farbwert setzen
0D04  3E FF      LD    A,FF         Kennzeichen für neue Farbwerte
0D06  32 FC B1   LD    (B1FC),A     setzen
0D09  C9         RET

**********************************  Adresse in Farbmatrix berechnen
                                    IN : Farbnummer
                                    OUT: Adresse des Farbwerts
0D0A  E6 1F      AND   1F              Farbnummer von 0..15
0D0C  C6 93      ADD   93                Basisadresse der Farbmatrix
0D0E  6F         LD    L,A               ($0D93)
0D0F  CE 0D      ADC   0D                addieren
0D11  95         SUB   L
0D12  67         LD    H,A
0D13  C9         RET

**********************************  SCR GET INK
                                    IN : A: Farbstiftnummer
                                    OUT: B: 1. Farbnummer
                                         C: 2. Farbnummer
0D14  E6 0F      AND   0F                Farbstiftnummer
0D16  3C         INC   A                 von 1..16
0D17  18 01      JR    0D1A

**********************************  SCR GET BORDER
                                    OUT: B: 1. Farbnummer
                                         C: 2. Farbnummer
0D19  AF         XOR   A              Farbstift 0 für BORDER
0D1A  CD 2F 0D   CALL  0D2F           Adressen in Farbtabellen holen
0D1D  1A         LD    A,(DE)         Wert aus 1. Farbsatz
0D1E  5E         LD    E,(HL)         Wert aus 2. Farbsatz
0D1F  CD 24 0D   CALL  0D24           1. Farbwert in Tabelle suchen
0D22  41         LD    B,C            1. Farbnummer nach B
0D23  7B         LD    A,E            2. Farbwert
0D24  0E 00      LD    C,00           Zähler für Farbnummer
0D26  21 93 0D   LD    HL,0D93        Adresse der Farbmatrix
0D29  BE         CP    (HL)           Farbwert gefunden ?
0D2A  C8         RET   Z              dann Farbnummer in C
0D2B  23         INC   HL             Zeiger in Tabelle
0D2C  0C         INC   C              und Farbnummer erhöhen
0D2D  18 FA      JR    0D29           weiter suchen

**********************************  Farbtabellenadressen holen
                                    IN : A: Farbstiftnummer
                                    OUT: DE: Zeiger auf 1. Farbwert
                                         HL: Zeiger auf 2. Farbwert
0D2F  5F         LD    E,A            Farbstiftnummer
0D30  16 00      LD    D,00           Nummer hi =0
0D32  21 EA B1   LD    HL,B1EA        Basisadressee d. 1. Farbsatzes
0D35  19         ADD   HL,DE          Nummer addieren
0D36  EB         EX    DE,HL          Adresse d. 1. Farbwerts n. DE
0D37  21 EF FF   LD    HL,FFEF          2. Farbsatz
0D3A  19         ADD   HL,DE            $11 Bytes früher
```

```
0D3B  C9          RET

***********************************  Farbwechsel-Event-Block einhängen
0D3C  21 FE B1    LD    HL,B1FE        Adresse des Event-Blocks
0D3F  E5          PUSH  HL             retten
0D40  CD 70 01    CALL  0170           Block aus Chain (FFC) löschen
0D43  CD 6D 0D    CALL  0D6D           Farbsätze wechs., Zähler init.
0D46  11 5B 0D    LD    DE,0D5B        Adresse der Event-Routine
0D49  06 81       LD    B,81           Priority Byte, async. Block
0D4B  E1          POP   HL             Adresse des Event-Blocks
0D4C  C3 63 01    JP    0163           KL NEW FRAME FLY, Block gener.

***********************************  Farbwechsel-Event-Block aushängen
0D4F  21 FE B1    LD    HL,B1FE        Adresse des Event-Blocks
0D52  CD 70 01    CALL  0170           Block aus Chain (FFC) löschen
0D55  CD 81 0D    CALL  0D81           Parameter des lfd. Farbsatzes
0D58  C3 86 07    JP    0786           MC CLEAR INKS, Farben init.

***********************************  Farbwechsel-Event-Routine
0D5B  21 FD B1    LD    HL,B1FD        Adresse des Zählers
0D5E  35          DEC   (HL)           Zähler herunterzählen
0D5F  28 0C       JR    Z,0D6D         =0 ? dann Farben wechseln
0D61  2B          DEC   HL
0D62  7E          LD    A,(HL)         Flag für neue Farbwerte
0D63  B7          OR    A              keine neuen Farbwerte ?
0D64  C8          RET   Z              dann zurück
0D65  CD 81 0D    CALL  0D81           Parameter d. lfd. Farbsatzes
0D68  CD 99 07    CALL  0799           Farben an Gate-Array übergeben
0D6B  18 0F       JR    0D7C           Flag für neue Farben löschen

***********************************  Farbsätze wechseln
0D6D  CD 81 0D    CALL  0D81           Param. d. lfd. Farbsatzes hol.
0D70  32 FD B1    LD    (B1FD),A       Zähler neu setzen
0D73  CD 99 07    CALL  0799           Farben an Gate-Array übergeben
0D76  21 FB B1    LD    HL,B1FB
0D79  7E          LD    A,(HL)            Flag für 1./2. Farbsatz
0D7A  2F          CPL                     invertieren
0D7B  77          LD    (HL),A            (anderen Farbsatz auswählen)
0D7C  AF          XOR   A              Flag für neue Farbwerte
0D7D  32 FC B1    LD    (B1FC),A       löschen
0D80  C9          RET

***********************************  Param. des lfd. Farbsatzes holen
                                     OUT: DE: Adresse der Farbtabelle
                                          A: Zählerwert
0D81  11 EA B1    LD    DE,B1EA        Adresse der 1. Farbwerttabelle
0D84  3A FB B1    LD    A,(B1FB)       Flag für 1./2. Farbsatz
0D87  B7          OR    A
0D88  3A D8 B1    LD    A,(B1D8)       Zählerwert für 1. Farbsatz
0D8B  C8          RET   Z              1. Farbsatz ? dann zurück
0D8C  11 D9 B1    LD    DE,B1D9        Adresse der 2. Farbwerttabelle
0D8F  3A D7 B1    LD    A,(B1D7)       Zählerwert für 2. Farbsatz
0D92  C9          RET

***********************************  Tabelle der Farbwerte entspr. Nr.
0D93  14 04 15 1C 18 1D 0C 05
0D9B  0D 16 06 17 1E 00 1F 0E
0DA3  07 0F 12 02 13 1A 19 1B
```

```
0DAB  0A 03 0B 01 08 09 10 11

*********************************** SCR FILL BOX
                                    IN : H: linke Grenze der Box
                                         L: obere Grenze
                                         D: rechte Grenze
                                         E: untere Grenze
                                         A: Füll-Byte (Farbmaske)
0DB3  4F          LD     C,A          Farb-Füll-Byte
0DB4  CD 95 0B    CALL   0B95         Param. aus Grenzen berechnen

*********************************** SCR FLOOD BOX
                                    IN : HL: Bildschirmadresse
                                         E: Zahl der Rasterzeilen
                                         D: Zahl der Bytes pro Zeile
                                         C: Füll-Byte (Farbmaske)
0DB7  E5          PUSH   HL           Bildschirmadresse retten
0DB8  7A          LD     A,D          Zahl der Bytes in einer Zeile
0DB9  CD E8 0E    CALL   0EE8         Übertrag zu RA-Bits ?
0DBC  30 09       JR     NC,0DC7      nein ? d. Speicherblock füllen
0DBE  42          LD     B,D          Zahl der Bytes pro Zeile
0DBF  71          LD     (HL),C       Byte in Video-RAM speichern
0DC0  CD F9 0B    CALL   0BF9         Zeiger auf nächstes Byte
0DC3  10 FA       DJNZ   0DBF         weitere Bytes in dies. Zeile ?
0DC5  18 10       JR     0DD7         nächste Zeile
0DC7  C5          PUSH   BC
0DC8  D5          PUSH   DE
0DC9  71          LD     (HL),C       1. Byte setzen
0DCA  15          DEC    D            restliche Zahl der Bytes
0DCB  28 08       JR     Z,0DD5       keine weiteren in dieser Zl. ?
0DCD  4A          LD     C,D          Zahl der Bytes
0DCE  06 00       LD     B,00         Zahl hi=0
0DD0  54          LD     D,H            Bildschirmadresse
0DD1  5D          LD     E,L            +1 als
0DD2  13          INC    DE             Zieladresse
0DD3  ED B0       LDIR                Speicherblock entspr. füllen
0DD5  D1          POP    DE
0DD6  C1          POP    BC
0DD7  E1          POP    HL           Bildschirmadresse
0DD8  CD 13 0C    CALL   0C13         Adresse d. nächst. Rasterzeile
0DDB  1D          DEC    E            Zähler für Rasterzeilen
0DDC  20 D9       JR     NZ,0DB7      weitere Rasterzeilen ?
0DDE  C9          RET

*********************************** SCR CHAR INVERT
                                    IN : H: Spalte
                                         L: Zeile
                                         B: Paper-Byte
                                         C: Pen-Byte
0DDF  78          LD     A,B          Pen-Byte
0DE0  A9          XOR    C            Unterschiede zu Paper-Byte
0DE1  4F          LD     C,A          nach C, gibt Invertier.-Byte
0DE2  CD 64 0B    CALL   0B64         Bildschirmadr., Bytes/Zeichen
0DE5  16 08       LD     D,08         Zähler für Rasterzeilen
0DE7  E5          PUSH   HL           Bildschirmadresse
0DE8  C5          PUSH   BC           Bytes/Zeichen, Invert.-Byte
0DE9  7E          LD     A,(HL)       Byte aus Bildschirm
0DEA  A9          XOR    C            Paper-/Pen-Pixels vertauschen
```

```
0DEB  77        LD    (HL),A      und Byte wieder speichern
0DEC  CD F9 0B  CALL  0BF9        Adresse des nächsten Bytes
0DEF  10 F8     DJNZ  0DE9        weitere Bytes ?
0DF1  C1        POP   BC          Bytes/Zeichen, Invert.-Byte
0DF2  E1        POP   HL          Bildschirmadresse
0DF3  CD 13 0C  CALL  0C13        Adr. der nächsten Rasterzeile
0DF6  15        DEC   D           Zähler für Rasterzeilen
0DF7  20 EE     JR    NZ,0DE7     weitere Rasterzeilen ?
0DF9  C9        RET

************************************ SCR HARDWARE ROLL
                                  IN : A: Paper-Byte
                                       B=0 für nach unten scrollen
                                       B=$FF für nach oben scrollen
0DFA  4F        LD    C,A         Paper-Byte
0DFB  C5        PUSH  BC          und Scroll-Flag retten
0DFC  11 D0 FF  LD    DE,FFD0     Offset f. unbenutzte Bytes
0DFF  06 30     LD    B,30        Zahl der Bytes im unben. Ber.
0E01  CD 24 0E  CALL  0E24        unben. Bereich m. Paper-B. fü.
0E04  C1        POP   BC          Scroll-Flag/Paper-Byte
0E05  CD BA 07  CALL  07BA        auf Strahlrücklauf warten
0E08  78        LD    A,B         Scroll-Flag
0E09  B7        OR    A
0E0A  20 0D     JR    NZ,0E19     nach oben scrollen ?
0E0C  11 B0 FF  LD    DE,FFB0     Bildsch.-Start 80 Bytes vorher
0E0F  CD 37 0E  CALL  0E37        Offset zu SCR OFFSET addieren
0E12  11 00 00  LD    DE,0000     Offset für 1. Bildschirmzeile
0E15  06 20     LD    B,20        Zahl der Bytes ($50-$30)
0E17  18 0B     JR    0E24        restl. 1. Bildschirmzl. füllen
0E19  11 50 00  LD    DE,0050     Bildsch.-Start 80 Bytes später
0E1C  CD 37 0E  CALL  0E37        Offset zu SCR OFFSET addieren
0E1F  11 B0 FF  LD    DE,FFB0     Offset f. letzte Bildschirmzl.
0E22  06 20     LD    B,20        Zahl der Bytes ($50-$30)

************************************ Textzeile teilweise füllen
                                  IN : C: Paper-Byte
                                       B: Zahl der zu füllenden
                                          Bytes pro Zeile
                                       DE: Offs. zu Bildsch.-Start
0E24  2A C9 B1  LD    HL,(B1C9)   SCR OFFSET
0E27  19        ADD   HL,DE       Offset addieren
0E28  7C        LD    A,H           eventuellen Übertrag
0E29  E6 07     AND   07            zu RA-Bits
0E2B  67        LD    H,A           wieder löschen
0E2C  3A CB B1  LD    A,(B1CB)    SCR BASE
0E2F  84        ADD   H           addieren
0E30  67        LD    H,A         gibt Bildschirmadresse
0E31  50        LD    D,B         Zahl der zu füllenden Byte/Zl.
0E32  1E 08     LD    E,08        Zahl der Rasterzeilen
0E34  C3 B7 0D  JP    0DB7        Bereich füllen

************************************ Offset zu SCR OFFSET addieren
                                  IN : DE: Offset
0E37  2A C9 B1  LD    HL,(B1C9)   SCR OFFSET
0E3A  19        ADD   HL,DE       Offset addieren
0E3B  C3 3C 0B  JP    0B3C        SCR OFFSET neu setzen
```

```
*********************************** SCR SOFTWARE ROLL
                                    IN : A: Paper-Byte
                                         B=0 für nach unten scrollen
                                         B=$FF für nach oben scrollen
                                         H: linke Window-Grenze
                                         L: obere Grenze
                                         D: rechte Grenze
                                         E: untere Grenze
0E3E  F5        PUSH  AF        Paper-Byte retten
0E3F  78        LD    A,B       Scroll-Flag
0E40  B7        OR    A
0E41  28 30     JR    Z,0E73    nach unten scrollen ?
0E43  E5        PUSH  HL        Grenzen links/oben
0E44  CD 95 0B  CALL  0B95      Window-Parameter berechnen
0E47  E3        EX    (SP),HL   Bildsch.-Adr. retten, Gr. zur.
0E48  2C        INC   L         nächste Zeile (Quellzeile)
0E49  CD 64 0B  CALL  0B64      Bildschirmadresse ber.
0E4C  4A        LD    C,D       Bytes pro Zeile im Window
0E4D  7B        LD    A,E       Rasterzeilen im Window
0E4E  D6 08     SUB   08        -8, 1 Textzl. nicht kopieren
0E50  47        LD    B,A       Zahl der zu kop. Rasterzeilen
0E51  28 17     JR    Z,0E6A    Null ? dann fertig
0E53  D1        POP   DE        Bildschirmadr. d. vorig. Zeile
0E54  CD BA 07  CALL  07BA      auf Strahlrücklauf warten
0E57  C5        PUSH  BC        Bytes pro Zeile/Rasterzeilenz.
0E58  E5        PUSH  HL        Quell-Bildschirmadresse
0E59  D5        PUSH  DE        Ziel-Bildschirmadresse
0E5A  CD A4 0E  CALL  0EA4      eine Rasterzeile kopieren
0E5D  E1        POP   HL        alte Zieladresse
0E5E  CD 13 0C  CALL  0C13      Adr. der nächsten Rasterzeile
0E61  EB        EX    DE,HL     neue Zieladresse nach DE
0E62  E1        POP   HL        alte Quelladresse
0E63  CD 13 0C  CALL  0C13      Adr. der nächsten Rasterzeile
0E66  C1        POP   BC        Bytes pro Zeile/Zeilenzähler
0E67  10 EE     DJNZ  0E57      weitere Rasterzeilen ?
0E69  D5        PUSH  DE        letzte Zieladresse

0E6A  E1        POP   HL        Adresse d. zu löschenden Zeile
0E6B  51        LD    D,C       Zahl der Bytes pro Zeile
0E6C  1E 08     LD    E,08      Zahl der Rasterzeilen
0E6E  F1        POP   AF        Paper-Byte
0E6F  4F        LD    C,A       nach C
0E70  C3 B7 0D  JP    0DB7      Textzeile löschen

0E73  E5        PUSH  HL        Grenzen für links/oben
0E74  D5        PUSH  DE        und rechts/unten retten
0E75  CD 95 0B  CALL  0B95      Window-Parameter berechnen
0E78  4A        LD    C,D       Bytes pro Zeile im Window
0E79  7B        LD    A,E       Rasterzeilenzahl im Window
0E7A  D6 08     SUB   08        -8, 1 Textzl. nicht kopieren
0E7C  47        LD    B,A       Zahl d. zu kop. Rasterzeilen
0E7D  D1        POP   DE        Windowgrenzen rechts/oben,
0E7E  E3        EX    (SP),HL   links/unten, Löschzeilenadr.
0E7F  28 E9     JR    Z,0E6A    keine Zeile zu kopieren ?
0E81  C5        PUSH  BC        Zahl d. Zeilen/Bytes pro Zeile
0E82  6B        LD    L,E       Grenze unten
0E83  54        LD    D,H       Grenze links
0E84  1C        INC   E         unterste Zeile +1
```

```
0E85  CD 64 0B    CALL  0B64       Bildschirmadr. f. unten/links
0E88  EB          EX    DE,HL      nach DE, als Quelladresse
0E89  CD 64 0B    CALL  0B64       Bildsch.-Adr. f. unten+1/links
0E8C  C1          POP   BC         Zahl d. Zeilen/Bytes pro Zeile
0E8D  CD BA 07    CALL  07BA       auf Strahlrücklauf warten
0E90  CD 2D 0C    CALL  0C2D       Adr. der vorigen Rasterzeile
0E93  E5          PUSH  HL         Zieladresse retten
0E94  EB          EX    DE,HL      Quelladresse nach HL
0E95  CD 2D 0C    CALL  0C2D       Adr. der vorigen Rasterzeile
0E98  E5          PUSH  HL         Quelladresse retten
0E99  C5          PUSH  BC
0E9A  CD A4 0E    CALL  0EA4       eine Rasterzeile kopieren
0E9D  C1          POP   BC
0E9E  D1          POP   DE         Quelladresse
0E9F  E1          POP   HL         Zieladresse
0EA0  10 EE       DJNZ  0E90       weitere Rasterzeilen ?
0EA2  18 C6       JR    0E6A       oberste Textzeile löschen

***********************************  Rasterzeile kopieren
                                  IN : HL: Adresse der Quellzeile
                                       DE: Adresse der Zielzeile
                                       C: zu kopierende Bytezahl
0EA4  06 00       LD    B,00        Bytezahl hi=0
0EA6  CD E6 0E    CALL  0EE6          Übertrag zu RA-Bits bei
0EA9  38 16       JR    C,0EC1        Zielzeile ?
0EAB  CD E6 0E    CALL  0EE6          kein Übertrag zu RA-Bits bei
0EAE  30 25       JR    NC,0ED5       Quellzeile ? d. direkt kop.
0EB0  C5          PUSH  BC          Zahl der Bytes retten
0EB1  AF          XOR   A             Zahl der Bytes bis zum näch-
0EB2  95          SUB   L             sten Übertrag aufs Hi-Byte
0EB3  4F          LD    C,A         als Länge
0EB4  ED B0       LDIR              Zeile bis dorthin kopieren
0EB6  C1          POP   BC          alte Zahl der Bytes
0EB7  2F          CPL                 Zahl der kopierten Bytes
0EB8  3C          INC   A             abziehen, gibt Zahl der noch
0EB9  81          ADD   C             zu kopierenden Bytes
0EBA  4F          LD    C,A         nach C
0EBB  7C          LD    A,H           RA-Bits der Quelladresse
0EBC  D6 08       SUB   08            wieder herstellen
0EBE  67          LD    H,A
0EBF  18 14       JR    0ED5        zweite Zeilenhälfte kopieren

0EC1  CD E6 0E    CALL  0EE6
0EC4  38 12       JR    C,0ED8
0EC6  C5          PUSH  BC            Zeile wie oben in zwei
0EC7  AF          XOR   A             Hälften kopieren
0EC8  93          SUB   E
0EC9  4F          LD    C,A
0ECA  ED B0       LDIR
0ECC  C1          POP   BC
0ECD  2F          CPL
0ECE  3C          INC   A
0ECF  81          ADD   C
0ED0  4F          LD    C,A
0ED1  7A          LD    A,D
0ED2  D6 08       SUB   08
0ED4  57          LD    D,A
0ED5  ED B0       LDIR
```

```
0ED7  C9          RET

0ED8  41          LD     B,C          Zahl der Bytes als Zähler
0ED9  7E          LD     A,(HL)         ein Byte
0EDA  12          LD     (DE),A         kopieren
0EDB  CD F9 0B    CALL   0BF9         nächstes Byte für Quelladresse
0EDE  EB          EX     DE,HL
0EDF  CD F9 0B    CALL   0BF9         nächstes Byte für Zieladresse
0EE2  EB          EX     DE,HL
0EE3  10 F4       DJNZ   0ED9         weitere Bytes zu kopieren ?
0EE5  C9          RET

***********************************  Test auf Übertrag zu RA-Bits
                                     IN : DE: Bildschirmadresse
                                           C: Zahl der Bytes ab dort
                                     OUT: HL: Bildschirmadresse
                                           CY=1 für Übertrag zu RA-Bits
0EE6  79          LD     A,C          Zahl der Bytes
0EE7  EB          EX     DE,HL        Bildschirmadresse nach DE
0EE8  3D          DEC    A            Offset zu letztem Byte
0EE9  85          ADD    L            zum Lo-Byte der Adresse add.
0EEA  D0          RET    NC           kein Übertrag zu Hi-Byte ?
0EEB  7C          LD     A,H          Hi-Byte
0EEC  E6 07       AND    07             Übertrag zu
0EEE  EE 07       XOR    07             RA-Bits ?
0EF0  C0          RET    NZ           nein ?
0EF1  37          SCF                 sonst CY=1 für Übertrag
0EF2  C9          RET

***********************************  SCR UNPACK
                                     IN : HL: Adr. d. gepackten Matrix
                                           DE: Zieladresse für
                                               ungepackte Matrix
0EF3  CD EC 0A    CALL   0AEC           Mode holen
0EF6  06 08       LD     B,08         Zähler für Zeilen
0EF8  38 31       JR     C,0F2B       Mode 0 ?
0EFA  28 06       JR     Z,0F02       Mode 1 ?
0EFC  01 08 00    LD     BC,0008      8 Bytes zu kopieren
0EFF  ED B0       LDIR                gepackte Matrix kopieren
0F01  C9          RET

0F02  4E          LD     C,(HL)       Byte aus gepackter Matrix
0F03  23          INC    HL           Zeiger auf nächstes Matrixbyte
0F04  E5          PUSH   HL           retten
0F05  C5          PUSH   BC           Rasterzeilen-Zähler
0F06  06 04       LD     B,04         4 Pixels pro ungepacktem Byte
0F08  21 CF B1    LD     HL,B1CF      Adr. d. Masken f. Pixelauswahl
0F0B  AF          XOR    A            ungepacktes Byte löschen
0F0C  CB 01       RLC    C            Pixel-Bit aus gepackter Matrix
0F0E  30 01       JR     NC,0F11      Pixel nicht gesetzt ?
0F10  B6          OR     (HL)         sonst entspr. Bits setzen
0F11  23          INC    HL           Zeiger auf nächste Maske
0F12  10 F8       DJNZ   0F0C         weitere Pixels im ungep. Byte?
0F14  12          LD     (DE),A       ungepacktes Byte speichern
0F15  13          INC    DE           nächstes Byte in ungep. Matrix
0F16  06 04       LD     B,04           nächste 4 Pixels aus
0F18  21 CF B1    LD     HL,B1CF        gepackter Matrix
0F1B  AF          XOR    A              entsprechend in ungepackte
```

```
0F1C  CB 01       RLC   C               Matrix übertragen
0F1E  30 01       JR    NC,0F21
0F20  B6          OR    (HL)
0F21  23          INC   HL
0F22  10 F8       DJNZ  0F1C
0F24  12          LD    (DE),A
0F25  13          INC   DE
0F26  C1          POP   BC
0F27  E1          POP   HL
0F28  10 D8       DJNZ  0F02          weitere Rasterzeilen ?
0F2A  C9          RET

0F2B  4E          LD    C,(HL)        Byte aus gepackter Matrix
0F2C  23          INC   HL            Zeiger auf nächstes Matrixbyte
0F2D  E5          PUSH  HL            retten
0F2E  C5          PUSH  BC            Rasterzeilen-Zähler
0F2F  06 04       LD    B,04          4 ungep. Bytes pro gep. Byte
0F31  AF          XOR   A             ungepacktes Byte löschen
0F32  21 CF B1    LD    HL,B1CF       Adr. d. Masken f. Pixelauswahl
0F35  CB 01       RLC   C             nächstes Bit aus ungep. Matrix
0F37  30 01       JR    NC,0F3A       Pixel nicht gesetzt ?
0F39  7E          LD    A,(HL)        Maske f. 1. Pixel ausgewählt
0F3A  23          INC   HL            Zeiger auf Maske f. 2. Pixel
0F3B  CB 01       RLC   C             nächstes Bit aus ungep. Matrix
0F3D  30 01       JR    NC,0F40       Pixel nicht gesetzt ?
0F3F  B6          OR    (HL)          Maske f. 2. Pixel ausgewählt
0F40  12          LD    (DE),A        ungepacktes Byte speichern
0F41  13          INC   DE            Zeiger auf nächst. ungep. Byte
0F42  10 ED       DJNZ  0F31          weitere ungep. Bytes ?
0F44  C1          POP   BC
0F45  E1          POP   HL
0F46  10 E3       DJNZ  0F2B          weitere Rasterzeilen ?
0F48  C9          RET

*********************************** SCR REPACK
                                    IN : H: Spalte
                                         L: Zeile
                                         A: Pen-Byte
                                         DE: Zieladresse für
                                             gepackte Matrix
0F49  4F          LD    C,A           Pen-Byte
0F4A  CD 64 0B    CALL  0B64          Bildschirmadr. berechnen
0F4D  CD EC 0A    CALL  0AEC          Mode holen
0F50  06 08       LD    B,08          8 Rasterzeilen
0F52  38 45       JR    C,0F99        Mode 0 ?
0F54  28 0B       JR    Z,0F61        Mode 1 ?
0F56  7E          LD    A,(HL)        Byte aus Bildschirm
0F57  A9          XOR   C             Pen-Pixels als 0-Bits
0F58  2F          CPL                 Pen-Pixels als 1-Bits
0F59  12          LD    (DE),A        Byte in gepackte Matrix sp.
0F5A  13          INC   DE            Zeiger in gepackter Matrix
0F5B  CD 13 0C    CALL  0C13          Adresse d. nächst. Rasterzeile
0F5E  10 F6       DJNZ  0F56          weitere Rasterzeilen ?
0F60  C9          RET

0F61  E5          PUSH  HL
0F62  D5          PUSH  DE
0F63  E5          PUSH  HL
```

```
0F64  7E          LD     A,(HL)      Byte aus Bildschirm
0F65  A9          XOR    C           Pen-Pixels als 0-Bits
0F66  21 CF B1    LD     HL,B1CF     Adr. d. Masken f. Pixelauswahl
0F69  16 04       LD     D,04        4 Pixels pro Byte
0F6B  F5          PUSH   AF          Pen-Pixel-Bits
0F6C  A6          AND    (HL)        Bits f. diesen Pixel isolieren
0F6D  20 01       JR     NZ,0F70     Pixel nicht in Pen-Farbe ?
0F6F  37          SCF                sonst 1-Bit in gepackte Matrix
0F70  CB 13       RL     E           Bit in gepackte Matrix rotier.
0F72  23          INC    HL          Adr. d. Maske f. nächst. Pixel
0F73  F1          POP    AF          Pen-Pixel-Bits
0F74  15          DEC    D
0F75  20 F4       JR     NZ,0F6B     weitere Pixels ?
0F77  E1          POP    HL
0F78  CD F9 0B    CALL   0BF9
0F7B  7E          LD     A,(HL)        nächste 4 Pixels in
0F7C  A9          XOR    C             nächstem Bildschirmbyte
0F7D  21 CF B1    LD     HL,B1CF       entsprechend packen
0F80  16 04       LD     D,04
0F82  F5          PUSH   AF
0F83  A6          AND    (HL)
0F84  20 01       JR     NZ,0F87
0F86  37          SCF
0F87  CB 13       RL     E
0F89  23          INC    HL
0F8A  F1          POP    AF
0F8B  15          DEC    D
0F8C  20 F4       JR     NZ,0F82
0F8E  E1          POP    HL
0F8F  73          LD     (HL),E      gepacktes Byte speichern
0F90  EB          EX     DE,HL
0F91  13          INC    DE          Zeiger in gepackte Matrix
0F92  E1          POP    HL          Bildschirmadresse
0F93  CD 13 0C    CALL   0C13        Adresse d. nächst. Rasterzeile
0F96  10 C9       DJNZ   0F61        weitere Rasterzeilen ?
0F98  C9          RET

0F99  E5          PUSH   HL
0F9A  D5          PUSH   DE
0F9B  16 04       LD     D,04        4 ungep. Bytes pro gep. Byte
0F9D  7E          LD     A,(HL)      Byte aus Bildschirm
0F9E  E5          PUSH   HL          Bildschirmadresse retten
0F9F  A9          XOR    C           Pen-Pixels als 0-Bits
0FA0  F5          PUSH   AF          Pen-Pixel-Bits
0FA1  21 CF B1    LD     HL,B1CF     Adr. d. Masken f. Pixelauswahl
0FA4  A6          AND    (HL)        Bits f. diesen Pixel isolieren
0FA5  20 01       JR     NZ,0FA8     Pixel nicht in Pen-Farbe ?
0FA7  37          SCF                sonst 1-Bit in gepackte Matrix
0FA8  CB 13       RL     E           Bit in gepackte Matrix rotier.
0FAA  F1          POP    AF          Pen-Pixel-Bits
0FAB  23          INC    HL          Adr. d. Maske f. nächst. Pixel
0FAC  A6          AND    (HL)          nächsten Pixel analog
0FAD  20 01       JR     NZ,0FB0       in Matrix-Byte
0FAF  37          SCF                  übertragen
0FB0  CB 13       RL     E
0FB2  E1          POP    HL
0FB3  CD F9 0B    CALL   0BF9        Adr. d. nächst. Bildsch.-Bytes
0FB6  15          DEC    D
```

```
0FB7  20 E4     JR    NZ,0F9D      weitere ungepackte Bytes ?
0FB9  E1        POP   HL
0FBA  73        LD    (HL),E       gepacktes Byte speichern
0FBB  EB        EX    DE,HL
0FBC  13        INC   DE           Zeiger auf nächst. Matrix-Byte
0FBD  E1        POP   HL           Bildschirmadresse
0FBE  CD 13 0C  CALL  0C13         Adr. der nächsten Rasterzeile
0FC1  10 D6     DJNZ  0F99         weitere Rasterzeilen ?
0FC3  C9        RET

*********************************** SCR HORIZONTAL
                                   IN : DE: X-Startkoordinate
                                        BC: X-Endkoordinate
                                        HL: Y-Koordinate
                                        A: Pen-Byte
0FC4  F5        PUSH  AF           Pen-Byte retten
0FC5  E5        PUSH  HL           Y-Koordinate retten
0FC6  7A        LD    A,D            Zweierkomplement
0FC7  2F        CPL                  der X-Startkoordinate
0FC8  67        LD    H,A            nach HL
0FC9  7B        LD    A,E
0FCA  2F        CPL
0FCB  6F        LD    L,A
0FCC  23        INC   HL
0FCD  09        ADD   HL,BC        X-Endk. minus X-Startk.
0FCE  23        INC   HL           +1= Zahl der X-Positionen
0FCF  E3        EX    (SP),HL      retten, Y-Koordinate zurück
0FD0  AF        XOR   A              Lo-Byte der X-Startkoord.
0FD1  93        SUB   E              negieren, für Abstand zum
0FD2  F5        PUSH  AF             nächsten vollen Byte retten
0FD3  CD A9 0B  CALL  0BA9         Bildsch.-Adr. und Maske ber.
0FD6  E5        PUSH  HL           Bildschirmadresse retten
0FD7  78        LD    A,B          signif. Bits f. Pixelstellung
0FD8  2F        CPL                invertieren, gibt
0FD9  6F        LD    L,A          signif. Bits f. Bildsch.-Adr.
0FDA  26 FF     LD    H,FF         im Hi-Byte alle Bits signifik.
0FDC  22 07 B2  LD    (B207),HL    sign. Bits als X-Offset/Byte
0FDF  E1        POP   HL           Bildschirmadresse
0FE0  F1        POP   AF           neg. Lo-Byte der X-Startkoord.
0FE1  A0        AND   B            sign. Bits f. Pixelst. isol.
0FE2  47        LD    B,A          gibt Pixelzahl bis näch. Byte
0FE3  28 45     JR    Z,102A       keine ? dann sofort byteweise
0FE5  E3        EX    (SP),HL      Bildsch.-Adr. r., X-Pos.-Zahl
0FE6  18 03     JR    0FEB
0FE8  1A        LD    A,(DE)       Bitmaske d. nächsten Pixels
0FE9  B1        OR    C            Bit(s) f. Pixel zusätzl. setz.
0FEA  4F        LD    C,A          neue Bitmaske
0FEB  2B        DEC   HL           X-Positionen-Zahl herunterz.
0FEC  7C        LD    A,H
0FED  B5        OR    L
0FEE  28 34     JR    Z,1024       keine weiteren X-Positionen ?
0FF0  13        INC   DE           Zeiger auf nächste Bitmaske
0FF1  10 F5     DJNZ  0FE8         weit. Pixels vor nächs. Byte ?
0FF3  EB        EX    DE,HL        Zahl der X-Positionen nach DE
0FF4  E1        POP   HL           Bildschirmadresse
0FF5  F1        POP   AF           Pen-Byte
0FF6  47        LD    B,A          nach B
0FF7  CD E8 BD  CALL  BDE8         SCR WRITE, Pixel(s) setzen
```

```
0FFA  CD F9 0B    CALL  0BF9          Adresse des nächsten Bytes
0FFD  E5          PUSH  HL            retten
0FFE  2A 07 B2    LD    HL,(B207)     neg. X-Offset zu nächstem Byte
1001  19          ADD   HL,DE         Offset v. X-Pos.-Zahl abziehen
1002  30 0C       JR    NC,1010       kein weiteres volles Byte ?
1004  EB          EX    DE,HL         Zahl der X-Positionen nach DE
1005  E1          POP   HL            Bildschirmadresse
1006  0E FF       LD    C,FF          Maske, alle Pixels ausgewählt
1008  CD E8 BD    CALL  BDE8          SCR WRITE, Pixels setzen
100B  CD F9 0B    CALL  0BF9          Adresse des nächsten Bytes
100E  18 ED       JR    0FFD          nächstes Byte
1010  7B          LD    A,E           Zahl der restl. X-Positionen
1011  B7          OR    A
1012  28 0E       JR    Z,1022        keine weiteren ?
1014  AF          XOR   A             Bitmaske f. kein Pixel
1015  21 CF B1    LD    HL,B1CF       Adr. d. Masken f. Pixelauswahl
1018  B6          OR    (HL)          Bit(s) f. Pixel setzen
1019  23          INC   HL            Adr. Bitmaske f. nächst. Pixel
101A  1D          DEC   E
101B  20 FB       JR    NZ,1018       weitere X-Positionen ?
101D  4F          LD    C,A           Bitmaske
101E  E1          POP   HL            Bildschirmadresse
101F  C3 E8 BD    JP    BDE8          SCR WRITE, Pixel(s) setzen

1022  E1          POP   HL            Bildschirmadresse
1023  C9          RET

1024  E1          POP   HL            Bildschirmadresse
1025  F1          POP   AF            Pen-Byte
1026  47          LD    B,A           nach B
1027  C3 E8 BD    JP    BDE8          SCR WRITE, Pixel(s) setzen
102A  D1          POP   DE            Zahl der X-Positionen
102B  F1          POP   AF            Pen-Byte
102C  47          LD    B,A           nach B
102D  18 CE       JR    0FFD          byteweise bearbeiten

***********************************  SCR VERTICAL
                                     IN : HL: Y-Startkoordinate
                                          BC: Y-Endkoordinate
                                          DE: X-Koordinate
                                          A: Pen-Byte
102F  F5          PUSH  AF            Pen-Byte retten
1030  E5          PUSH  HL            Y-Startkoordinate retten
1031  7C          LD    A,H             Zweierkomplement
1032  2F          CPL                   der Y-Startkoordinate
1033  67          LD    H,A             nach HL
1034  7D          LD    A,L
1035  2F          CPL
1036  6F          LD    L,A
1037  23          INC   HL
1038  09          ADD   HL,BC         Y-Endkoord. minus Y-Startk.
1039  23          INC   HL            +1 =Zahl der Y-Positionen
103A  E3          EX    (SP),HL       retten, Y-Startk. zurück
103B  CD A9 0B    CALL  0BA9          Bildschirmadr. und Maske holen
103E  D1          POP   DE            Zahl der Y-Positionen
103F  F1          POP   AF            Pen-Byte
1040  47          LD    B,A           nach B
1041  CD E8 BD    CALL  BDE8          SCR WRITE, Pixel setzen
```

```
1044  CD 2D 0C      CALL   0C2D       Adr. der vorigen Rasterzeile
1047  1B            DEC    DE         Zahl der Y-Positionen
1048  7A            LD     A,D          weitere Y-Positionen
1049  B3            OR     E            (Rasterzeilen)
104A  20 F5         JR     NZ,1041      zu bearbeiten ?
104C  C9            RET

********************************** Default-Farbwerte
104D  04 04                           BORDER-Werte
104F  0A 13 0C 0B 14 15 0D 06
1057  1E 1F 07 12 19 04 17 04         INK-Werte
105F  04 0A 13 0C 0B 14 15 0D
1067  06 1E 1F 07 12 19 0A 07

106F  C7            RST    00
1070  C7            RST    00
1071  C7            RST    00
1072  C7            RST    00
1073  C7            RST    00
1074  C7            RST    00
1075  C7            RST    00
1076  C7            RST    00
1077  C7            RST    00
```

```
------------------------ TEXT SCREEN (TXT) -------------------------

**********************************         TXT INITIALIZE
1078  CD 88 10     CALL   1088            TXT RESET
107B  AF           XOR    A               Flag für User-Matrix
107C  32 95 B2     LD     (B295),A        löschen
107F  21 01 00     LD     HL,0001         PAPER 0, PEN 1
1082  CD 3D 11     CALL   113D            Default-Parameter setzen
1085  C3 A3 10     JP     10A3            alle Windows initialisieren

**********************************         TXT RESET
1088  21 91 10     LD     HL,1091         Zeiger auf Block
108B  CD 8A 0A     CALL   0A8A            Indirections kopieren
108E  C3 5B 14     JP     145B            Steuerzeichen-Tab. init.
1091  0F                                  Zahl der zu kopierenden Bytes
1092  CD BD                               Zieladresse im RAM
1094  C3 63 12     JP     1263            TXT DRAW CURSOR
1097  C3 63 12     JP     1263            TXT UNDRAW CURSOR
109A  C3 4A 13     JP     134A            TXT WRITE CHAR
109D  C3 C0 13     JP     13C0            TXT UNWRITE
10A0  C3 0C 14     JP     140C            TXT OUT ACTION

**********************************         alle Windows entspr. akt. Window
10A3  3E 08        LD     A,08            Zähler für Windows
10A5  11 0D B2     LD     DE,B20D         Zeiger auf Params f. Window 0
10A8  21 85 B2     LD     HL,B285         Zeiger auf akt. Window-Params
10AB  01 0F 00     LD     BC,000F         Länge der Window-Parameter
10AE  ED B0        LDIR                   akt. Window kopieren
10B0  3D           DEC    A
10B1  20 F5        JR     NZ,10A8         weitere Windows ?
10B3  32 0C B2     LD     (B20C),A        Window 0 auswählen
10B6  C9           RET

**********************************         alle Farben decodieren
                                           OUT: C: akt. Window
10B7  3A 0C B2     LD     A,(B20C)        aktuelle Window-Nummer
10BA  4F           LD     C,A             nach C retten
10BB  06 08        LD     B,08            Zähler für 8 Windows
10BD  78           LD     A,B             akt. Zählerwert
10BE  3D           DEC    A               Window von 0..7
10BF  CD E8 10     CALL   10E8            Window auswählen
10C2  CD D0 BD     CALL   BDD0            Cursor ausschalten
10C5  CD C3 12     CALL   12C3            Paper-Wert holen
10C8  32 90 B2     LD     (B290),A        und speichern (uncodiert!)
10CB  CD BD 12     CALL   12BD            Pen-Wert holen
10CE  32 8F B2     LD     (B28F),A        und speichern (uncodiert!)
10D1  10 EA        DJNZ   10BD            weitere Windows ?
10D3  79           LD     A,C             alte Window-Nummer
10D4  C9           RET

**********************************         alle Farben cod., Wind. auf Def.
                                           IN : A: anschließend auszuwäh-
                                                   lende Window-Nummer
10D5  4F           LD     C,A             Window-Nummer retten
10D6  06 08        LD     B,08            Zähler für 8 Windows
10D8  78           LD     A,B             akt. Zählerwert
10D9  3D           DEC    A               Window von 0..7
10DA  CD E8 10     CALL   10E8            Window auswählen
```

```
10DD  C5          PUSH  BC
10DE  2A 8F B2    LD    HL,(B28F)     Paper-/Pen-Werte (uncodiert!)
10E1  CD 3D 11    CALL  113D          Default-Parameter setzen
10E4  C1          POP   BC
10E5  10 F1       DJNZ  10D8          weitere Windows ?
10E7  79          LD    A,C           neue Window-Nummer

*********************************** TXT STR SELECT
                                    IN : A: neue Window-Nummer
                                    OUT: A: alte Window-Nummer
10E8  E6 07       AND   07            Window-Nummer von 0..7
10EA  21 0C B2    LD    HL,B20C       akt. Window-Nummer
10ED  BE          CP    (HL)          Window bereits ausgewählt ?
10EE  C8          RET   Z             dann fertig
10EF  C5          PUSH  BC
10F0  D5          PUSH  DE
10F1  4E          LD    C,(HL)        alte Window-Nummer
10F2  77          LD    (HL),A        neue Window-Nummer setzen
10F3  47          LD    B,A           neue Nummer
10F4  79          LD    A,C           alte Nummer
10F5  CD 2A 11    CALL  112A          Adr. der lfd./alten Parameter
10F8  CD 22 11    CALL  1122          laufende in alte Params kop.
10FB  78          LD    A,B           neue Window-Nummer
10FC  CD 2A 11    CALL  112A          Adr. der lfd./neuen Parameter
10FF  EB          EX    DE,HL         vertauschen
1100  CD 22 11    CALL  1122          neue in laufende Params kop.
1103  79          LD    A,C           alte Window-Nummer
1104  D1          POP   DE
1105  C1          POP   BC
1106  C9          RET

*********************************** TXT SWAP STREAMS
                                    IN : B: 1. Window-Nummer
                                         C: 2. Window-Nummer
1107  3A 0C B2    LD    A,(B20C)      aktuelle Window-Nummer
110A  F5          PUSH  AF            retten
110B  79          LD    A,C           2. Window-Nr.
110C  CD E8 10    CALL  10E8          als akt. setzen, Params kop.
110F  78          LD    A,B           1. Window-Nr.
1110  32 0C B2    LD    (B20C),A      als akt. Nr. f. Zurückkopieren
1113  CD 2A 11    CALL  112A          Adresse der Params d. 1. Nr.
1116  D5          PUSH  DE            retten
1117  79          LD    A,C           2. Window-Nr.
1118  CD 2A 11    CALL  112A          Adresse der Params d. 2. Nr.
111B  E1          POP   HL            Adresse f. 1. Nr. als Quelle
111C  CD 22 11    CALL  1122          Params v. 1. in 2. Window kop.
111F  F1          POP   AF            alte akt. Window-Nummer
1120  18 C6       JR    10E8          wieder setzen

*********************************** Window-Parameter kopieren
                                    IN : HL: Quelladresse
                                         DE: Zieladresse
1122  C5          PUSH  BC
1123  01 0F 00    LD    BC,000F       Länge der Params eines Windows
1126  ED B0       LDIR                Parameter kopieren
1128  C1          POP   BC
1129  C9          RET
```

```
***********************************     Adr. der Window-Parameter holen
                                        IN : A: Window-Nummer
                                        OUT: DE: Adr. der zugeh. Params
                                             HL: Adr. der lfd. Parameter
112A  E6 07       AND    07               Window-Nummer von 0..7
112C  5F          LD     E,A
112D  87          ADD    A
112E  87          ADD    A                  mal 15, da 15 Parameter-
112F  87          ADD    A                  Bytes pro Window
1130  87          ADD    A
1131  93          SUB    E
1132  C6 0D       ADD    0D               $B20D (Basisadresse der
1134  5F          LD     E,A              Window-Parameter-Tabelle
1135  CE B2       ADC    B2               addieren
1137  93          SUB    E
1138  57          LD     D,A
1139  21 85 B2    LD     HL,B285          Adr. der laufenden Parameter
113C  C9          RET

***********************************     Window-Default-Parameter setzen
                                        IN : H: Paper-Wert (uncodiert!)
                                             L: Pen-Wert (uncodiert!)
113D  EB          EX     DE,HL            Paper-/Pen-Wert
113E  3E 03       LD     A,03               Flag für Cursor OFF und
1140  32 8D B2    LD     (B28D),A           DISABLED setzen
1143  7A          LD     A,D              Paper-Wert
1144  CD AE 12    CALL   12AE             setzen
1147  7B          LD     A,E              Pen-Wert
1148  CD A9 12    CALL   12A9             setzen
114B  AF          XOR    A                  Flag für TAGOFF
114C  CD A7 13    CALL   13A7               setzen
114F  CD 7A 13    CALL   137A             FORCE-Hintergrund-Modus
1152  21 00 00    LD     HL,0000          Grenzen links/oben
1155  11 7F 7F    LD     DE,7F7F          Grenzen rechts/unten
1158  CD 0C 12    CALL   120C             Windowgrenzen setzen
115B  C3 51 14    JP     1451             VDU-Flag auf ENABLED

***********************************     TXT SET COLUMN
                                        IN : A: Spalte
115E  3D          DEC    A                relativer Spaltenwert +1
115F  21 89 B2    LD     HL,B289            + linke Windowgrenze
1162  86          ADD    (HL)               gibt absolute Position
1163  2A 85 B2    LD     HL,(B285)        Cursorposition
1166  67          LD     H,A              Spalte neu setzen
1167  18 0E       JR     1177             Cursor neu setzen

***********************************     TXT SET ROW
                                        IN : A: Zeile
1169  3D          DEC    A                relativer Zeilenwert -1
116A  21 88 B2    LD     HL,B288            + obere Windowgrenze
116D  86          ADD    (HL)               gibt absolute Position
116E  2A 85 B2    LD     HL,(B285)        Cursorposition
1171  6F          LD     L,A              Zeile neu setzen
1172  18 03       JR     1177             Cursor neu setzen
```

```
*********************************** TXT SET CURSOR
                                    IN : H: Spalte
                                         L: Zeile
1174  CD 8A 11     CALL   118A         relative in absolute Position
1177  CD D0 BD     CALL   BDD0         TXT UNDRAW CURSOR
117A  22 85 B2     LD     (B285),HL    neue Cursorposition setzen
117D  C3 CD BD     JP     BDCD         TXT DRAW CURSOR

*********************************** TXT GET CURSOR
                                    OUT: H: Spalte
                                         L: Zeile
                                         A: Scrolling-Zähler
1180  2A 85 B2     LD     HL,(B285)    Cursorposition
1183  CD 97 11     CALL   1197         in relative Position wandeln
1186  3A 8C B2     LD     A,(B28C)     Scrolling-Zähler
1189  C9           RET

*********************************** relative in absolute Pos. wandeln
                                    IN/OUT: H: Spalte
                                            L: Zeile
118A  3A 88 B2     LD     A,(B288)     obere Windowgrenze
118D  3D           DEC    A            -1 = Offset
118E  85           ADD    L            zu Zeile addieren
118F  6F           LD     L,A          als absolute Zeile
1190  3A 89 B2     LD     A,(B289)     linke Windowgrenze
1193  3D           DEC    A            -1 = Offset
1194  84           ADD    H            zu Spalte addieren
1195  67           LD     H,A          als absolute Spalte
1196  C9           RET

*********************************** absolute in relative Pos. wandeln
                                    IN/OUT: H: Spalte
                                            L: Zeile
1197  3A 88 B2     LD     A,(B288)     obere Windowgrenze
119A  95           SUB    L            - Zeile
119B  2F           CPL                   Zweierkomplement,
119C  3C           INC    A              Zeile-obere Grenze
119D  3C           INC    A            +1
119E  6F           LD     L,A          gibt relative Zeile
119F  3A 89 B2     LD     A,(B289)       mit rechter Windowgrenze
11A2  94           SUB    H              und Spalte
11A3  2F           CPL                   ebenso
11A4  3C           INC    A              verfahren
11A5  3C           INC    A
11A6  67           LD     H,A
11A7  C9           RET
*********************************** Cursor invert., Position prüfen
                                    OUT: H: Cursorspalte
                                         L: Cursorzeile
11A8  CD D0 BD     CALL   BDD0         TXT UNDRAW CURSOR

*********************************** Cursorpos. prüfen, ggf. scrollen
                                    OUT: H: Cursorspalte
                                         L: Cursorzeile
11AB  2A 85 B2     LD     HL,(B285)    Cursorposition
11AE  CD DA 11     CALL   11DA         innerh. Windowgrenzen bringen
11B1  22 85 B2     LD     (B285),HL    und wieder setzen
11B4  D8           RET    C            war Zeile innerhalb Grenzen ?
```

```
11B5  E5         PUSH   HL           Cursorposition retten
11B6  21 8C B2   LD     HL,B28C      Adr. des Scrolling-Zählers
11B9  78         LD     A,B          Scrolling-Flag
11BA  87         ADD    A              A=1 für nach unten, A=$FF
11BB  3C         INC    A              für nach oben scrollen
11BC  86         ADD    (HL)         zu Zähler addieren
11BD  77         LD     (HL),A       Scrolling-Zähler wieder setzen
11BE  CD 56 12   CALL   1256         Window-Parameter holen
11C1  3A 90 B2   LD     A,(B290)     Paper-Byte f. zu löschende Zl.
11C4  F5         PUSH   AF           Paper-Byte und Flag retten
11C5  DC 3E 0E   CALL   C,0E3E       ggf. softwaremäßig scrollen
11C8  F1         POP    AF
11C9  D4 FA 0D   CALL   NC,0DFA      ggf. hardwaremäßig scrollen
11CC  E1         POP    HL           Cursorposition zurück
11CD  C9         RET

**********************************  TXT VALIDATE
                                    IN/OUT: H: Spalte
                                               L: Zeile
                                    OUT: CY=1 für Position o.k.
                                           CY=0 für Scrolling
11CE  CD 8A 11   CALL   118A           relative in absolute Position
11D1  CD DA 11   CALL   11DA           innerhalb Grenzen bringen
11D4  F5         PUSH   AF             Scroll-Flag retten
11D5  CD 97 11   CALL   1197           absolute in relative Position
11D8  F1         POP    AF             Scroll-Flag
11D9  C9         RET

**********************************  Position in Grenzen forcieren
                                    IN/OUT: H: Spalte
                                               L: Zeile
                                    OUT: CY=0, wenn Scrolling nötig
                                           dann:
                                           B=0 für nach unten scrollen
                                           B=$FF für nach oben scrollen
11DA  3A 8B B2   LD     A,(B28B)       rechte Grenze
11DD  BC         CP     H              Spalte <= rechte Grenze ?
11DE  F2 E6 11   JP     P,11E6         dann o.k.
11E1  3A 89 B2   LD     A,(B289)       sonst linke Grenze
11E4  67         LD     H,A            als Spalte
11E5  2C         INC    L              Zeile erhöhen
11E6  3A 89 B2   LD     A,(B289)       linke Grenze
11E9  3D         DEC    A
11EA  BC         CP     H              Spalte > linke Grenze-1 ?
11EB  FA F3 11   JP     M,11F3         dann o.k.
11EE  3A 8B B2   LD     A,(B28B)       sonst rechte Grenze
11F1  67         LD     H,A            als Spalte
11F2  2D         DEC    L              Zeile erniedrigen
11F3  3A 88 B2   LD     A,(B288)       obere Grenze
11F6  3D         DEC    A
11F7  BD         CP     L              Zeile <= obere Grenze-1 ?
11F8  F2 06 12   JP     P,1206         dann korrigieren
11FB  3A 8A B2   LD     A,(B28A)       untere Grenze
11FE  BD         CP     L              Zeile <= untere Grenze
11FF  37         SCF                   CY=1 für innerhalb Grenzen
1200  F0         RET    P              dann o.k., fertig
1201  6F         LD     L,A            sonst untere Grenze als Zeile
1202  06 FF      LD     B,FF           Flag für nach oben scrollen
```

```
1204  B7           OR     A            CY=0 für Scrolling nötig
1205  C9           RET
1206  3C           INC    A            obere Grenze
1207  6F           LD     L,A          als Zeile setzen
1208  06 00        LD     B,00         Flag für nach unten scrollen
120A  B7           OR     A            CY=0 für Scrolling nötig
120B  C9           RET

**********************************   TXT WIN ENABLE
                                     IN : H,D: Spaltengrenzen
                                          L,E: Zeilengrenzen
120C  CD 57 0B     CALL   0B57         maximale Grenzen holen
120F  7C           LD     A,H          linke Grenze
1210  CD 44 12     CALL   1244         in zulässige Grenzen bringen
1213  67           LD     H,A          und wieder setzen
1214  7A           LD     A,D          rechte Grenze
1215  CD 44 12     CALL   1244         in zulässige Grenzen bringen
1218  57           LD     D,A          und wieder setzen
1219  BC           CP     H
121A  30 02        JR     NC,121E      rechte >= linke Grenze ?
121C  54           LD     D,H          sonst linke als rechte
121D  67           LD     H,A          und rechte als linke Grenze
121E  7D           LD     A,L
121F  CD 4D 12     CALL   124D
1222  6F           LD     L,A            ebenso mit oberer und
1223  7B           LD     A,E            unterer Grenze verfahren
1224  CD 4D 12     CALL   124D
1227  5F           LD     E,A
1228  BD           CP     L
1229  30 02        JR     NC,122D
122B  5D           LD     E,L
122C  6F           LD     L,A
122D  22 88 B2     LD     (B288),HL    Grenzen links/oben
1230  ED 53 8A B2  LD     (B28A),DE    und rechts/unten setzen
1234  7C           LD     A,H            Grenze links
1235  B5           OR     L              oder oben <>0 ?
1236  20 06        JR     NZ,123E      dann Software-Scrolling
1238  7A           LD     A,D          rechte Grenze
1239  A8           XOR    B            nicht maximale Grenze ?
123A  20 02        JR     NZ,123E      dann Software-Scrolling
123C  7B           LD     A,E          obere Grenze
123D  A9           XOR    C            = Maximalgrenze ?
123E  32 87 B2     LD     (B287),A     Flag f. Soft-/Hardware-Scroll.
1241  C3 77 11     JP     1177         Cursor nach links/oben setzen


**********************************   Spaltengrenze in zuläss. Bereich
                                     IN/OUT: A: Window-Spaltengrenze
                                             B: maximale Spaltengrenze
1244  B7           OR     A
1245  F2 49 12     JP     P,1249       Grenze >=0 ?
1248  AF           XOR    A            sonst Null als Grenze
1249  B8           CP     B            Grenze < Maximalwert ?
124A  D8           RET    C            dann zurück
124B  78           LD     A,B          sonst Maximalwert als Grenze
124C  C9           RET
```

```
***********************************  Zeilengrenze in zuläss. Bereich
                                     IN/OUT: A: Window-Zeilengrenze
                                             C: maximale Zeilengrenze
124D  B7           OR    A
124E  F2 52 12     JP    P,1252         Grenze >=0 ?
1251  AF           XOR   A              sonst Null als Grenze
1252  B9           CP    C              Grenze < Maximalwert ?
1253  D8           RET   C              dann zurück
1254  79           LD    A,C            sonst Maximalwert als Grenze
1255  C9           RET

***********************************  TXT GET WINDOW
                                     OUT: H: linke Grenze
                                          L: obere Grenze
                                          D: rechte Grenze
                                          E: untere Grenze
                                          CY=0 für Hardware-Scrolling
1256  2A 88 B2     LD    HL,(B288)      Grenzen links/oben
1259  ED 5B 8A B2  LD    DE,(B28A)      Grenzen rechts/unten
125D  3A 87 B2     LD    A,(B287)       Scrolling-Flag
1260  C6 FF        ADD   FF             CY=0 bei Hardware-Scrolling
1262  C9           RET

***********************************  TXT DRAW/UNDRAW CURSOR
1263  3A 8D B2     LD    A,(B28D)       Cursor-Flag
1266  B7           OR    A              Cursor OFF oder DISABLED ?
1267  C0           RET   NZ             dann zurück

***********************************  TXT PLACE/REMOVE CURSOR
1268  C5           PUSH  BC
1269  D5           PUSH  DE
126A  E5           PUSH  HL
126B  CD AB 11     CALL  11AB           Cursorpos. prf., ggf. scrollen
126E  ED 4B 8F B2  LD    BC,(B28F)      Pen- und Paper-Byte
1272  CD DF 0D     CALL  0DDF           Cursor invertieren
1275  E1           POP   HL
1276  D1           POP   DE
1277  C1           POP   BC
1278  C9           RET

***********************************  TXT CUR ON
1279  F5           PUSH  AF
127A  3E FD        LD    A,FD           Bit 1, ON/OFF-Flag
127C  CD 8B 12     CALL  128B           Flag auf ON setzen
127F  F1           POP   AF
1280  C9           RET

***********************************  TXT CUR OFF
1281  F5           PUSH  AF
1282  3E 02        LD    A,02           Bit 1, ON/OFF-Flag
1284  CD 9C 12     CALL  129C           Flag auf OFF setzen
1287  F1           POP   AF
1288  C9           RET

***********************************  TXT CUR ENABLE
1289  3E FE        LD    A,FE           Bit 0, ENABLE/DISABLE-Flag
128B  F5           PUSH  AF
128C  CD D0 BD     CALL  BDD0           TXT UNDRAW CURSOR, ggf. inv.
```

```
128F  F1        POP   AF
1290  E5        PUSH  HL
1291  21 8D B2  LD    HL,B28D        Adr. des Cursor-Flags
1294  A6        AND   (HL)           entsprechendes Bit löschen
1295  77        LD    (HL),A         und Flags wieder setzen
1296  E1        POP   HL
1297  C3 CD BD  JP    BDCD           TXT DRAW CURSOR

*********************************** TXT CUR DISABLE
129A  3E 01     LD    A,01           Bit 0, ENABLE/DISABLE-Flag
129C  F5        PUSH  AF
129D  CD D0 BD  CALL  BDD0           TXT UNDRAW CURSOR
12A0  F1        POP   AF
12A1  E5        PUSH  HL
12A2  21 8D B2  LD    HL,B28D        Adr. des Cursor-Flags
12A5  B6        OR    (HL)           entsprechendes Bit setzen
12A6  77        LD    (HL),A         und Flags wieder setzen
12A7  E1        POP   HL
12A8  C9        RET

*********************************** TXT SET PEN
                                    IN : A: Farbstift-Nummer
12A9  21 8F B2  LD    HL,B28F        Adresse f. akt. Pen-Wert
12AC  18 03     JR    12B1           Wert setzen

*********************************** TXT SET PAPER
                                    IN : A: Farbstift-Nummer
12AE  21 90 B2  LD    HL,B290        Adresse f. akt. Paper-Wert
12B1  F5        PUSH  AF             neuen Wert retten
12B2  CD D0 BD  CALL  BDD0           TXT UNDRAW CURSOR
12B5  F1        POP   AF             neuer Wert
12B6  CD 86 0C  CALL  0C86           zugeh. Farbmaske berechnen
12B9  77        LD    (HL),A         und Wert setzen
12BA  C3 CD BD  JP    BDCD           TXT DRAW CURSOR

*********************************** TXT GET PEN
                                    OUT: A: Farbstift-Nummer
12BD  3A 8F B2  LD    A,(B28F)       Pen-Farbmaske
12C0  C3 A0 0C  JP    0CA0           zugeh. Farbstift-Nr. berechnen

*********************************** TXT GET PAPER
                                    OUT: A: Farbstift-Nummer
12C3  3A 90 B2  LD    A,(B290)       Paper-Farbmaske
12C6  C3 A0 0C  JP    0CA0           zugeh. Farbstift-Nr. berechnen

*********************************** TXT INVERSE
12C9  2A 8F B2  LD    HL,(B28F)      Paper- und Pen-Byte
12CC  7C        LD    A,H
12CD  65        LD    H,L              vertauschen
12CE  6F        LD    L,A
12CF  22 8F B2  LD    (B28F),HL      und wieder setzen
12D2  C9        RET
```

```
***********************************    TXT GET MATRIX
                                       IN : A: Nr. des Zeichens
                                       OUT: HL: Adresse der Matrix
                                            CY=1, wenn User-Matrix
12D3  D5          PUSH   DE
12D4  5F          LD     E,A              Nr. des Zeichens
12D5  CD 2A 13    CALL   132A             Parameter d. User-Matrix holen
12D8  30 09       JR     NC,12E3          keine User-Matrix ?
12DA  57          LD     D,A              Nr. d. 1. User-Matrix-Zeichens
12DB  7B          LD     A,E              akt. Zeichen
12DC  92          SUB    D                minus 1. Zeichen
12DD  3F          CCF
12DE  30 03       JR     NC,12E3          Zeichen nicht in User-Matrix ?
12E0  5F          LD     E,A              Nr. innerhalb User-Matrix
12E1  18 03       JR     12E6             Adresse berechnen
12E3  21 00 38    LD     HL,3800          Basisadresse der ROM-Matrizen
12E6  F5          PUSH   AF               Flag f. User-Matrix retten
12E7  16 00       LD     D,00             Nr. des Zeichens hi =0
12E9  EB          EX     DE,HL
12EA  29          ADD    HL,HL              Nummer mal 8
12EB  29          ADD    HL,HL
12EC  29          ADD    HL,HL
12ED  19          ADD    HL,DE            plus Basisadresse
12EE  F1          POP    AF               Flag f. User-Matrix
12EF  D1          POP    DE
12F0  C9          RET

***********************************    TXT SET MATRIX
                                       IN : A: Zeichen
                                            HL: Adresse der Matrix
12F1  EB          EX     DE,HL            Adresse d. neuen Matrix n. DE
12F2  CD D3 12    CALL   12D3             Zieladresse f. Matrix holen
12F5  D0          RET    NC               keine User-Matrix ?
12F6  EB          EX     DE,HL            Zieladresse nach DE
12F7  01 08 00    LD     BC,0008          Länge der Matrix
12FA  ED B0       LDIR                    Matrix kopieren
12FC  C9          RET

***********************************    TXT SET M TABLE
                                       IN : D=0, wenn User-Matrix
                                            E: 1. Zeichen in User-Matrix
                                            HL: Zeiger auf User-Matrix
                                       OUT: CY=1, wenn alte User-Matrix
                                            HL: Adr. d. alten User-Matr.
                                            A: 1. Zeichen in alter Matr.
12FD  E5          PUSH   HL               Zeiger auf User-Matrix retten
12FE  7A          LD     A,D              Flag für User-Matrix
12FF  B7          OR     A
1300  16 00       LD     D,00             Kennz. für keine User-Matrix
1302  20 19       JR     NZ,131D          keine User-Matrix ?
1304  15          DEC    D                sonst $FF für User-Matrix
1305  D5          PUSH   DE               Flag/1.Zeichen retten
1306  4B          LD     C,E              1. Zeichen
1307  EB          EX     DE,HL            Zeiger für Matrix nach DE
1308  79          LD     A,C              akt. Zeichen
1309  CD D3 12    CALL   12D3             Adr. (aus alter Matrix) n. HL
130C  7C          LD     A,H
130D  AA          XOR    D
```

```
130E  20 04        JR    NZ,1314       Quell- und Zieladresse für
1310  7D           LD    A,L           Zeichenmatrix gleich ?
1311  AB           XOR   E
1312  28 08        JR    Z,131C       dann nicht weiter kopieren
1314  C5           PUSH  BC           Nr. des akt. Zeichens
1315  CD F7 12     CALL  12F7         Matrix in neue Matrix kopieren
1318  C1           POP   BC           Nr. des akt. Zeichens
1319  0C           INC   C            nächstes Zeichen
131A  20 EC        JR    NZ,1308      weitere Zeichen ?
131C  D1           POP   DE           Flag/Nr. des 1. Zeichens
131D  CD 2A 13     CALL  132A         Params der alten Matrix holen
1320  ED 53 94 B2  LD    (B294),DE    Flag/1. Zeichen setzen
1324  D1           POP   DE           Adresse der User-Matrix
1325  ED 53 96 B2  LD    (B296),DE    setzen
1329  C9           RET

***********************************  TXT GET M TABLE
                                     OUT: CY=1, wenn User-Matrix
                                          HL: Adresse d. User-Matrix
                                          A: Nr. des 1. Zeichens
132A  2A 94 B2     LD    HL,(B294)    Flag/Nr. des 1. Zeichens
132D  7C           LD    A,H
132E  0F           RRCA               Flag ins Carry
132F  7D           LD    A,L          Nr. des 1. Zeichens
1330  2A 96 B2     LD    HL,(B296)    Adresse der User-Matrix
1333  C9           RET

***********************************  TXT WR CHAR
                                     IN : A: Zeichen
1334  47           LD    B,A          Zeichen retten
1335  3A 8E B2     LD    A,(B28E)     VDU-Flag
1338  B7           OR    A
1339  C8           RET   Z            disabled ? dann zurück
133A  C5           PUSH  BC           Zeichen retten
133B  CD A8 11     CALL  11A8         Cursor invert., Position prf.
133E  24           INC   H            Cursorspalte erhöhen
133F  22 85 B2     LD    (B285),HL    Cursorpos. neu setzen
1342  25           DEC   H            alte Cursorspalte
1343  F1           POP   AF           auszugebendes Zeichen
1344  CD D3 BD     CALL  BDD3         TXT WRITE CHAR, Zeichen ausg.
1347  C3 CD BD     JP    BDCD         Cursor wieder an, Pos. prüfen

***********************************  TXT WRITE CHAR
                                     IN : A: Zeichen
                                          H: Spalte
                                          L: Zeile
134A  E5           PUSH  HL           Spalte/Zeile retten
134B  CD D3 12     CALL  12D3         Adr. der Zeichenmatrix n. HL
134E  11 98 B2     LD    DE,B298      Adr. f. ungepackte Matrix
1351  D5           PUSH  DE
1352  CD F3 0E     CALL  0EF3         Matrix auf Bildschirmformat
1355  D1           POP   DE           Adresse der ungepackten Matrix
1356  E1           POP   HL           Text-Spalte/Zeile
1357  CD 64 0B     CALL  0B64         Bildschirmadr./Bytezahl holen
135A  0E 08        LD    C,08         Zahl der Rasterzeilen
135C  C5           PUSH  BC           Zeilenzähler/Bytezahl retten
135D  E5           PUSH  HL           Bildschirmadr. der 1. Spalte
135E  C5           PUSH  BC           Zeilen- und Bytezähler retten
```

```
135F  D5          PUSH   DE            Zeiger in ungepackte Matrix
1360  EB          EX     DE,HL         nach HL
1361  4E          LD     C,(HL)        Byte aus ungepackter Matrix
1362  CD 76 13    CALL   1376          in Bildschirm setzen
1365  CD F9 0B    CALL   0BF9          Adresse des nächsten Bytes
1368  D1          POP    DE
1369  13          INC    DE            nächstes Matrix-Byte
136A  C1          POP    BC            Bytezähler
136B  10 F1       DJNZ   135E          weitere Bytes in dies. Zeile ?
136D  E1          POP    HL            Adr. der 1. Rasterspalte d. Z.
136E  CD 13 0C    CALL   0C13          Adr. d. nächsten Rasterzeile
1371  C1          POP    BC            Zeilenzähler/Bytezahl
1372  0D          DEC    C
1373  20 E7       JR     NZ,135C       weitere Rasterzeilen ?
1375  C9          RET

***********************************  Textzeichen-Byte auf Bildschirm
                                     IN : DE: Bildschirmadresse
                                           C: Textzeichen-Matrix-Byte
1376  2A 91 B2    LD     HL,(B291)     Adr. entspr. Hintergrund-Modus
1379  E9          JP     (HL)          entsprechende Routine ausführ.

***********************************  TXT SET BACK
                                     IN : A=0 für Pixel-Kopie
                                          entsprechend Matrix
                                          A=1 für OR-Verknüpfung
137A  21 91 13    LD     HL,1391       Adresse für Kopie-Modus
137D  B7          OR     A
137E  28 03       JR     Z,1383        Modus ausgewählt ?
1380  21 9F 13    LD     HL,139F       sonst Adresse für OR-Modus
1383  22 91 B2    LD     (B291),HL     setzen
1386  C9          RET

***********************************  TXT GET BACK
                                     OUT: Z=1 für Pixel-Kopie
                                          entsprechend Matrix
                                          Z=0 für OR-Verknüpfung
1387  2A 91 B2    LD     HL,(B291)     Adresse entsprechend Modus
138A  11 6F EC    LD     DE,EC6F
138D  19          ADD    HL,DE           Kopie-Modus (Adresse $1391)?
138E  7C          LD     A,H             dann Z=1
138F  B5          OR     L
1390  C9          RET

***********************************  Byte setzen, Kopie der Matrix
                                     IN : DE: Bildschirmadresse
                                           C: Textzeichen-Matrix-Byte
1391  2A 8F B2    LD     HL,(B28F)     Paper- und Pen-Maske
1394  79          LD     A,C           Matrix-Byte
1395  2F          CPL                    0-Bits aus Matrix
1396  A4          AND    H               entsprechend Paper-Maske
1397  47          LD     B,A           Paper-Pixels retten
1398  79          LD     A,C           Matrix-Byte
1399  A5          AND    L             1-Bits entspr. Pen-Maske
139A  B0          OR     B             Paper-Bits dazu
139B  0E FF       LD     C,FF          alle Bits verändern
139D  18 03       JR     13A2          Byte setzen
```

```
**********************************     Byte setzen, OR-Verknüpfung
                                       IN : DE: Bildschirmadresse
                                            C: Textzeichen-Matrix-Byte
139F  3A 8F B2    LD    A,(B28F)         Pen-Byte
13A2  47          LD    B,A              nach B
13A3  EB          EX    DE,HL            Bildschirmadresse nach HL
13A4  C3 6B 0C    JP    0C6B             Pixel(s) zusätzlich setzen

**********************************     TXT SET GRAPHIC
                                       IN : A=0 für Zeichen an Textpos.
                                            A<>0 f. Graphik-Positionen
13A7  32 93 B2    LD    (B293),A
13AA  C9          RET

**********************************     TXT RD CHAR
                                       OUT: CY=1 f. Zeich. identifiziert
                                            Z=1 für Space
                                            A: Zeichen
13AB  E5          PUSH  HL
13AC  D5          PUSH  DE
13AD  C5          PUSH  BC
13AE  CD D0 BD    CALL  BDD0             TXT UNDRAW CURSOR
13B1  2A 85 B2    LD    HL,(B285)        Cursorposition
13B4  CD D6 BD    CALL  BDD6             TXT UNWRITE, Zeichen lesen
13B7  F5          PUSH  AF               Zeichen und Flags retten
13B8  CD CD BD    CALL  BDCD             TXT UNDRAW CURSOR
13BB  F1          POP   AF
13BC  C1          POP   BC
13BD  D1          POP   DE
13BE  E1          POP   HL
13BF  C9          RET

**********************************     TXT UNWRITE
                                       IN : H: Spalte
                                            L: Zeile
                                       OUT: CY=1 f. Zeich. identifiziert
                                            Z=1 für Space
                                            A: Zeichen
13C0  3A 8F B2    LD    A,(B28F)         Pen-Byte
13C3  11 98 B2    LD    DE,B298          Adr. f. gepackte Matrix
13C6  E5          PUSH  HL               Spalte/Zeile
13C7  D5          PUSH  DE               Adresse f. Matrix
13C8  CD 49 0F    CALL  0F49             Matrix aus Bildsch. packen
13CB  CD E3 13    CALL  13E3             gepackte Matrix suchen
13CE  D1          POP   DE               Adresse der Matrix
13CF  E1          POP   HL               Spalte/Zeile
13D0  30 01       JR    NC,13D3          Matrix nicht gefunden ?
13D2  C0          RET   NZ               kein Space ? dann zurück
13D3  3A 90 B2    LD    A,(B290)         Paper-Byte
13D6  D5          PUSH  DE
13D7  CD 49 0F    CALL  0F49             Matrix aus Bildschirm packen
13DA  D1          POP   DE
13DB  06 08       LD    B,08             8 Rasterzeilen
13DD  1A          LD    A,(DE)           Byte aus Matrix
13DE  2F          CPL                    invertieren
13DF  12          LD    (DE),A           und wieder speichern
13E0  13          INC   DE
13E1  10 FA       DJNZ  13DD             weitere Rasterzeilen ?
```

```
***********************************  gepackte Matrix suchen
                                     IN : gep. Matrix ab $B298
                                     OUT: CY=1, wenn gefunden
                                          Z=1, wenn Space
                                          Z: Zeichen
13E3  0E 00      LD    C,00            Zeichenzähler
13E5  79         LD    A,C             akt. Zeichen
13E6  CD D3 12   CALL  12D3            Adresse der Matrix holen
13E9  11 98 B2   LD    DE,B298         Zeiger auf gesuchte Matrix
13EC  06 08      LD    B,08            8 Rasterzeilen
13EE  1A         LD    A,(DE)          Byte aus gesuchter Matrix
13EF  BE         CP    (HL)            mit gegebener Matrix vergl.
13F0  20 09      JR    NZ,13FB         ungleich ? dann näch. Zeichen
13F2  23         INC   HL
13F3  13         INC   DE
13F4  10 F8      DJNZ  13EE            weitere Rasterzeilen/Bytes ?
13F6  79         LD    A,C             gefundenes Zeichen
13F7  FE 20      CP    20              Z=1, wenn Space
13F9  37         SCF                   CY=1 für gefunden
13FA  C9         RET
13FB  0C         INC   C               Nr. des Zeichens erhöhen
13FC  20 E7      JR    NZ,13E5         weitere Zeichen ?
13FE  AF         XOR   A               sonst CY=0 für nicht gefunden
13FF  C9         RET

***********************************  TXT OUTPUT
                                     IN : A: Zeichen
1400  F5         PUSH  AF
1401  C5         PUSH  BC
1402  D5         PUSH  DE
1403  E5         PUSH  HL
1404  CD D9 BD   CALL  BDD9            TXT OUT ACTION
1407  E1         POP   HL
1408  D1         POP   DE
1409  C1         POP   BC
140A  F1         POP   AF
140B  C9         RET

***********************************  TXT OUT ACTION
                                     IN : A: Zeichen
140C  4F         LD    C,A             Zeichen
140D  3A 93 B2   LD    A,(B293)        Graphik-(TAG-)Flag
1410  B7         OR    A
1411  79         LD    A,C             Zeichen
1412  C2 45 19   JP    NZ,1945         gesetzt ? dann an Graphikpos.
1415  21 B8 B2   LD    HL,B2B8         Adr. d. Control-Buffer-Zähler
1418  46         LD    B,(HL)          Zahl der Zeichen im Buffer
1419  78         LD    A,B
141A  FE 0A      CP    0A              schon maximale Länge ?
141C  30 28      JR    NC,1446         dann Buffer löschen
141E  B7         OR    A
141F  20 06      JR    NZ,1427         schon Zeichen im Buffer ?
1421  79         LD    A,C             Zeichen
1422  FE 20      CP    20              kein Steuerzeichen ?
1424  D2 34 13   JP    NC,1334         dann direkt ausgeben
1427  04         INC   B               Zahl d. Zeichen im Buffer erh.
1428  70         LD    (HL),B          Zeichenzahl neu setzen
1429  58         LD    E,B             Länge des Buffers
```

```
142A  16 00       LD    D,00        Länge hi=0
142C  19          ADD   HL,DE       zu Basisadresse addieren
142D  71          LD    (HL),C      Zeichen speichern
142E  3A B9 B2    LD    A,(B2B9)    zugeh. Steuerzeichen
1431  5F          LD    E,A         nach E
1432  21 C3 B2    LD    HL,B2C3     Adr. d. Steuerzeichentabelle
1435  19          ADD   HL,DE         Nr. des Steuerzeichens
1436  19          ADD   HL,DE         3 mal addieren, da 3 Bytes
1437  19          ADD   HL,DE         pro Eintrag
1438  7E          LD    A,(HL)      Zahl der benötigten Zeichen
1439  B8          CP    B           noch nicht genug Zeichen ?
143A  D0          RET   NC          dann zurück
143B  23          INC   HL          Zeiger auf Ausführadresse
143C  5E          LD    E,(HL)        Adresse
143D  23          INC   HL            nach DE
143E  56          LD    D,(HL)
143F  21 B9 B2    LD    HL,B2B9     Zeiger auf Control Buffer
1442  79          LD    A,C         zuletzt übergebenes Zeichen
1443  CD 16 00    CALL  0016        Steuerzeichen-Routine ausführ.
1446  AF          XOR   A             Zahl der Zeichen im Control-
1447  32 B8 B2    LD    (B2B8),A      Buffer=0
144A  C9          RET

************************************    TXT VDU DISABLE
144B  CD 9A 12    CALL  129A          TXT CUR DISABLE
144E  AF          XOR   A             Null für disabled
144F  18 05       JR    1456

************************************    TXT VDU ENABLE
1451  CD 89 12    CALL  1289          TXT CUR ENABLE
1454  3E FF       LD    A,FF          $FF für enabled
1456  32 8E B2    LD    (B28E),A      VDU-Flag setzen
1459  18 EB       JR    1446          Control-Buffer-Zähler lösch.

************************************    Steuerzeichentabelle init.
145B  AF          XOR   A             Control-Buffer-Zähler
145C  32 B8 B2    LD    (B2B8),A      löschen
145F  21 6B 14    LD    HL,146B       Zeiger a. Default-Werte im ROM
1462  11 C3 B2    LD    DE,B2C3       Zeiger auf RAM-Bereich
1465  01 60 00    LD    BC,0060       $20 Steuerzeichen, 3 Bytes/Z.
1468  ED B0       LDIR                Steuerzeichentabelle kopieren
146A  C9          RET


************************************    Steuerzeichentabelle (Default)
                                        ( Zahl der auf das Steuerzeichen
                                          folgenden Bytes sowie
                                          Ausführadresse )
146B  00 E2 14    14E2, CHR$(0)       Zeiger auf RET
146E  01 34 13    1334, CHR$(1)       TXT WR CHAR (direkte Ausgabe)
1471  00 9A 12    129A, CHR$(2)       TXT CUR DISABLE
1474  00 89 12    1289, CHR$(3)       TXT CUR ENABLE
1477  01 CA 0A    0ACA, CHR$(4)       SCR SET MODE
147A  01 45 19    1945, CHR$(5)       GRA WR CHAR
147D  00 51 14    1451, CHR$(6)       TXT VDU ENABLE
1480  00 D8 14    14D8, CHR$(7)       Ton ausgeben
1483  00 0A 15    150A, CHR$(8)       Cursor left
1486  00 0F 15    150F, CHR$(9)       Cursor right
```

```
1489  00 14 15      1514, CHR$(10)      Cursor down/Linefeed
148C  00 19 15      1519, CHR$(11)      Cursor up
148F  00 40 15      1540, CHR$(12)      TXT CLEAR WINDOW
1492  00 30 15      1530, CHR$(13)      Carriage return
1495  01 AE 12      12AE, CHR$(14)      TXT SET PAPER
1498  01 A9 12      12A9, CHR$(15)      TXT SET PEN
149B  00 4F 15      154F, CHR$(16)      Zeichen unter Cursor löschen
149E  00 8E 15      158E, CHR$(17)      Zeile bis Cursor löschen
14A1  00 84 15      1584, CHR$(18)      Zeile ab Cursor löschen
14A4  00 6D 15      156D, CHR$(19)      Bildschirm bis Cursor löschen
14A7  00 56 15      1556, CHR$(20)      Bildschirm ab Cursor löschen
14AA  00 4B 14      144B, CHR$(21)      TXT VDU DISABLE
14AD  01 E3 14      14E3, CHR$(22)      Hintergrundmodus auswählen
14B0  01 49 0C      0C49, CHR$(23)      SCR ACCESS
14B3  00 C9 12      12C9, CHR$(24)      TXT INVERSE
14B6  09 04 15      1504, CHR$(25)      Zeichenmatrix setzen
14B9  04 F8 14      14F8, CHR$(26)      Window definieren
14BC  00 E2 14      14E2, CHR$(27)      Zeiger auf RET
14BF  03 E8 14      14E8, CHR$(28)      Farbstift definieren (INK)
14C2  02 F1 14      14F1, CHR$(29)      Rand setzen (BORDER)
14C5  00 2A 15      152A, CHR$(30)      Cursor home (links oben)
14C8  02 38 15      1538, CHR$(31)      Cursorposition setzen (LOCATE)


*********************************** TXT GET CONTROLS
                                    OUT: HL: Adr. Steuerzeichentab.
14CB  21 C3 B2      LD      HL,B2C3
14CE  C9            RET

*********************************** SOUND QUEUE-Parameter für CHR$(7)
14CF  87 00 00 5A 00 00 0B 14
14D7  00

*********************************** CHR$(7), Ton erzeugen
14D8  DD E5         PUSH    IX
14DA  21 CF 14      LD      HL,14CF        Zeiger auf Parameter
14DD  CD 9F 1F      CALL    1F9F           SOUND QUEUE, Ton erzeugen
14E0  DD E1         POP     IX
14E2  C9            RET

*********************************** CHR$(22), Hintergrundmodus setzen
14E3  0F            RRCA                   Flag ins Carry
14E4  9F            SBC     A              Flag nach 00/$FF
14E5  C3 7A 13      JP      137A           TXT SET BACK, Modus setzen

*********************************** CHR$(28), Farbstift setzen
14E8  23            INC     HL
14E9  7E            LD      A,(HL)         Farbstiftnr.
14EA  23            INC     HL
14EB  46            LD      B,(HL)         1. Farbnr.
14EC  23            INC     HL
14ED  4E            LD      C,(HL)         2. Farbnr.
14EE  C3 EC 0C      JP      0CEC           SCR SET INK, Farbstift setzen

*********************************** CHR$(29), Rand setzen
14F1  23            INC     HL
14F2  46            LD      B,(HL)         1. Farbnr.
14F3  23            INC     HL
```

```
14F4  4E            LD    C,(HL)        2. Farbnr.
14F5  C3 F1 0C      JP    0CF1          SCR SET BORDER, Rand setzen

************************************ CHR$(26), Window definieren
14F8  23            INC   HL
14F9  56            LD    D,(HL)          Spalten-
14FA  23            INC   HL
14FB  7E            LD    A,(HL)          grenzen
14FC  23            INC   HL
14FD  5E            LD    E,(HL)          und Zeilen-
14FE  23            INC   HL
14FF  6E            LD    L,(HL)          grenzen laden
1500  67            LD    H,A
1501  C3 0C 12      JP    120C          TXT WIN ENABLE, Grenzen setzen

************************************ CHR$(25), Zeichenmatrix def.
1504  23            INC   HL
1505  7E            LD    A,(HL)        Nr. des Zeichens
1506  23            INC   HL            Adresse der Matrix
1507  C3 F1 12      JP    12F1          TXT SET MATRIX, Matrix zuordn.

************************************ CHR$(8), Cursor left
150A  11 00 FF      LD    DE,FF00       Offset f. Spalte erniedrigen
150D  18 0D         JR    151C

************************************ CHR$(9), Cursor right
150F  11 00 01      LD    DE,0100       Offset f. Spalte erhöhen
1512  18 08         JR    151C

************************************ CHR$(10), Cursor down/Linefeed
1514  11 01 00      LD    DE,0001       Offset f. Zeile erhöhen
1517  18 03         JR    151C

************************************ CHR$(11), Cursor up
1519  11 FF 00      LD    DE,00FF       Offset f. Zeile erniedrigen
151C  D5            PUSH  DE
151D  CD A8 11      CALL  11A8          Cursor invert., Position prf.
1520  D1            POP   DE
1521  7D            LD    A,L
1522  83            ADD   E               Offset zu Zeile,
1523  6F            LD    L,A
1524  7C            LD    A,H
1525  82            ADD   D               2. Offset zu Spalte addieren
1526  67            LD    H,A
1527  C3 7A 11      JP    117A          Pos. setzen, Cursor wieder an

************************************ CHR$(30), Cursor home
152A  2A 88 B2      LD    HL,(B288)     Windowgrenzen links/oben
152D  C3 77 11      JP    1177          absolute Cursorposition setzen


************************************ CHR$(13), Carriage return
1530  CD A8 11      CALL  11A8          Cursor invert., Pos. nach HL
1533  3A 89 B2      LD    A,(B289)      linke Window-Grenze
1536  18 EE         JR    1526          als Spalte setzen
```

```
***********************************  CHR$(31), Cursorposition setzen
1538  23          INC    HL
1539  56          LD     D,(HL)        neue Cursorspalte
153A  23          INC    HL
153B  5E          LD     E,(HL)        und -zeile laden
153C  EB          EX     DE,HL         neue Position nach HL
153D  C3 74 11    JP     1174          Cursorposition neu setzen

***********************************  TXT CLEAR WINDOW, CHR$(12)
1540  CD D0 BD    CALL   BDD0          TXT UNDRAW CURSOR
1543  2A 88 B2    LD     HL,(B288)     Window-Grenzen links/oben
1546  22 85 B2    LD     (B285),HL     als Cursorposition
1549  ED 5B 8A B2 LD     DE,(B28A)     Window-Grenzen rechts/unten
154D  18 48       JR     1597          links/oben-rechts/unten lösch.

***********************************  CHR$(16), Cursor-Zeichen löschen
154F  CD A8 11    CALL   11A8          Cursor invert., Position n. HL
1552  54          LD     D,H             als Endposition
1553  5D          LD     E,L             nach DE
1554  18 41       JR     1597          Zeichen löschen

***********************************  CHR$(20), Bildsch. ab Cursor lö.
1556  CD 84 15    CALL   1584          Zeile ab Cursor löschen
1559  2A 88 B2    LD     HL,(B288)     Window-Grenzen links/oben
155C  ED 5B 8A B2 LD     DE,(B28A)     Window-Grenzen rechts/unten
1560  3A 85 B2    LD     A,(B285)      Cursorzeile
1563  6F          LD     L,A             +1 als
1564  2C          INC    L               Startzeile
1565  BB          CP     E             Startzeile < Endzeile ?
1566  3A 90 B2    LD     A,(B290)      Paper-Byte
1569  DC B3 0D    CALL   C,0DB3        dann Bereich füllen
156C  C9          RET

***********************************  CHR$(19), Bildsch. bis Cursor lö.
156D  CD 8E 15    CALL   158E          Zeile bis Cursor löschen
1570  2A 88 B2    LD     HL,(B288)     Window-Grenzen links/oben
1573  3A 8B B2    LD     A,(B28B)      rechte Grenze
1576  57          LD     D,A           als End-Spalte
1577  3A 85 B2    LD     A,(B285)      Cursorzeile
157A  3D          DEC    A             -1
157B  5F          LD     E,A           als End-Zeile
157C  BD          CP     L             Endzeile >= Startzeile ?
157D  3A 90 B2    LD     A,(B290)      Paper-Byte
1580  D4 B3 0D    CALL   NC,0DB3       dann Bereich füllen
1583  C9          RET

***********************************  CHR$(18), Zeile ab Cursor löschen
1584  CD A8 11    CALL   11A8          Cursor invert., Position n. HL
1587  5D          LD     E,L           Cursorzeile als Start- & Endz.
1588  3A 8B B2    LD     A,(B28B)      rechte Windowgrenze
158B  57          LD     D,A           als End-Spalte
158C  18 09       JR     1597          Bereich löschen

***********************************  CHR$(17), Zeile bis Cursor lösch.
158E  CD A8 11    CALL   11A8          Cursor invert., Position n. HL
1591  EB          EX     DE,HL         nach DE als Endposition
1592  6B          LD     L,E           Cursorzeile als Startzeile
1593  3A 89 B2    LD     A,(B289)      linke Window-Grenze
```

```
1596   67          LD     H,A          als Start-Spalte
1597   3A 90 B2    LD     A,(B290)     Paper-Byte
159A   CD B3 0D    CALL   0DB3         SCR FILL BOX, Bereich füllen
159D   CD CD BD    CALL   BDCD         TXT DRAW CURSOR
15A0   C9          RET
15A1   C7          RST    00
15A2   C7          RST    00
15A3   C7          RST    00
15A4   C7          RST    00
15A5   C7          RST    00
15A6   C7          RST    00
15A7   C7          RST    00
15A8   C7          RST    00
15A9   C7          RST    00
15AA   C7          RST    00
15AB   C7          RST    00
15AC   C7          RST    00
15AD   C7          RST    00
15AE   C7          RST    00
15AF   C7          RST    00
```

```
---------------------- GRAPHICS SCREEN (GRA) ----------------------

***********************************   GRA INITIALIZE
15B0  CD DF 15     CALL   15DF          GRA RESET
15B3  21 01 00     LD     HL,0001       PAPER 0, PEN 1
15B6  7C           LD     A,H           0
15B7  CD FD 17     CALL   17FD          als Paper-Farbstift
15BA  7D           LD     A,L           1
15BB  CD F6 17     CALL   17F6          als Pen-Farbstift setzen
15BE  21 00 00     LD     HL,0000
15C1  54           LD     D,H             Origin = 0/0
15C2  5D           LD     E,L
15C3  CD 04 16     CALL   1604          Origin setzen
15C6  11 00 80     LD     DE,8000       min. Wert
15C9  21 FF 7F     LD     HL,7FFF       max. Wert
15CC  E5           PUSH   HL              Werte
15CD  D5           PUSH   DE              retten
15CE  CD 34 17     CALL   1734          Windowgrenzen links/rechts
15D1  E1           POP    HL
15D2  D1           POP    DE
15D3  C3 79 17     JP     1779          und oben/unten setzen

***********************************   Graphik-Pen und Paper decodieren
                                      OUT: H: Paper
                                           L: Pen
15D6  CD 0A 18     CALL   180A          GRA GET PAPER
15D9  67           LD     H,A           Paper-Farbstift
15DA  CD 04 18     CALL   1804          GRA GET PEN
15DD  6F           LD     L,A           Pen-Farbstift
15DE  C9           RET

***********************************   GRA RESET
15DF  21 E5 15     LD     HL,15E5       Adresse der Rom-Tabelle
15E2  C3 8A 0A     JP     0A8A          Indirections kopieren
15E5  09                                Zahl der zu kopierenden Bytes
15E6  DC BD                             Zieladresse in Ram
15E8  C3 16 18     JP     1816          GRA PLOT
15EB  C3 2A 18     JP     182A          GRA TEST
15EE  C3 3C 18     JP     183C          GRA LINE

***********************************   GRA MOVE RELATIVE
                                      IN : DE: relative X-Koordinate
                                           HL: relative Y-Koordinate
15F1  CD 57 16     CALL   1657          relative in absolute Koord.

***********************************   GRA MOVE ABSOLUTE
                                      IN : DE: X-Koordinate
                                           HL: Y-Koordinate
15F4  ED 53 2C B3  LD     (B32C),DE       als Graphikcursorposition
15F8  22 2E B3     LD     (B32E),HL       setzen
15FB  C9           RET

***********************************   GRA ASK CURSOR
                                      OUT: DE: X-Koordinate
                                           HL: Y-Koordinate
15FC  ED 5B 2C B3  LD     DE,(B32C)     X-
1600  2A 2E B3     LD     HL,(B32E)     und Y-Koordiante laden
1603  C9           RET
```

```
**********************************  GRA SET ORIGIN
                                    IN : DE: X-Koordinate
                                         HL: Y-Koordinate
1604  ED 53 28 B3  LD     (B328),DE      Origin
1608  22 2A B3     LD     (B32A),HL      setzen
160B  11 00 00     LD     DE,0000
160E  62           LD     H,D            Graphikcursorposition 0/0
160F  6B           LD     L,E
1610  18 E2        JR     15F4         setzen

**********************************  GRA GET ORIGIN
                                    OUT: DE: X-Koordinate
                                         HL: Y-Koordinate
1612  ED 5B 28 B3  LD     DE,(B328)    X-
1616  2A 2A B3     LD     HL,(B32A)    und Y-Origin laden
1619  C9           RET

**********************************  reale Cursorkoordinaten holen
                                    OUT: DE: X-Koordinate
                                         HL: Y-Koordinate
161A  CD FC 15     CALL   15FC         Graphikcursorposition holen

**********************************  Cursor setzen, reale Koord. holen
                                    IN/OUT: DE: X-Koordinate
                                            HL: Y-Koordinate
161D  CD F4 15     CALL   15F4         Cursorposition neu setzen
1620  E5           PUSH   HL           Y-Koordinate retten
1621  CD EC 0A     CALL   0AEC         Mode holen
1624  2F           CPL                   A=3,1,0 für Mode 0,1,2
1625  C6 01        ADD    01             (Maske für nicht
1627  CE 02        ADC    02             signifikante Bits in Koord.)
1629  26 00        LD     H,00           Maske
162B  6F           LD     L,A            nach HL
162C  CB 7A        BIT    7,D
162E  28 03        JR     Z,1633       X-Koordiante positiv ?
1630  EB           EX     DE,HL          Maske addieren, damit
1631  19           ADD    HL,DE          Wegrundung der nicht signif.
1632  EB           EX     DE,HL          Bits zu Null hin erfolgt
1633  2F           CPL                 nicht signifikante Bits
1634  A3           AND    E            aus X-Koordinate löschen
1635  5F           LD     E,A
1636  7D           LD     A,L          nicht signifikante Bits
1637  2A 28 B3     LD     HL,(B328)    X-Origin-Koordinate
163A  19           ADD    HL,DE        zu X-Koordinate addieren
163B  0F           RRCA                Mode 0 oder Mode 1 ?
163C  DC 74 17     CALL   C,1774       dann Koordinate halbieren
163F  0F           RRCA                Mode 0 ?
1640  DC 74 17     CALL   C,1774       dann Koord. nochmals halbieren
1643  D1           POP    DE           Y-Koordinate
1644  E5           PUSH   HL           reale X-Koordinate retten
1645  7A           LD     A,D
1646  07           RLCA
1647  30 01        JR     NC,164A      Y-Koordinate positiv ?
1649  13           INC    DE           Ausgleich, zu Null hin runden
164A  7B           LD     A,E            Bit 0 löschen, da
164B  E6 FE        AND    FE             nicht signifikant
164D  5F           LD     E,A
164E  2A 2A B3     LD     HL,(B32A)    Y-Origin
```

```
1651  19          ADD   HL,DE       zu Y-Koordinate addieren
1652  CD 74 17    CALL  1774        Koordinate halbieren
1655  D1          POP   DE          reale X-Koordinate zurück
1656  C9          RET

**********************************  Cursor-relative in absol. Koord.
                                    IN/OUT: DE: X-Koordinate
                                            HL: Y-Koordinate
1657  E5          PUSH  HL
1658  2A 2C B3    LD    HL,(B32C)     X-Cursorposition
165B  19          ADD   HL,DE         zu X-Koordinate addieren
165C  D1          POP   DE
165D  E5          PUSH  HL
165E  2A 2E B3    LD    HL,(B32E)     Y-Cursorposition
1661  19          ADD   HL,DE         zu Y-Koordinate addieren
1662  D1          POP   DE
1663  C9          RET

**********************************  Test, ob Koordinaten f. vertikale
                                    Linie innerhalb Grenzen sind
                                    IN/OUT: HL: Y-Startkoordinate
                                            BC: Y-Endkoordinate
                                            DE: X-Koordinate
                                            (reale Koordinaten!)
                                    OUT: CY=0 f. außerhalb Grenzen
1664  D5          PUSH  DE
1665  E5          PUSH  HL
1666  2A 30 B3    LD    HL,(B330)
1669  2B          DEC   HL
166A  B7          OR    A
166B  ED 52       SBC   HL,DE
166D  F2 AC 16    JP    P,16AC
1670  2A 32 B3    LD    HL,(B332)
1673  B7          OR    A
1674  ED 52       SBC   HL,DE          Test analog zu Test für
1676  FA AC 16    JP    M,16AC         horizontale Linie ($16B0)
1679  D1          POP   DE             durchführen
167A  2A 34 B3    LD    HL,(B334)
167D  B7          OR    A
167E  ED 52       SBC   HL,DE
1680  FA AD 16    JP    M,16AD
1683  2A 36 B3    LD    HL,(B336)
1686  2B          DEC   HL
1687  B7          OR    A
1688  ED 52       SBC   HL,DE
168A  FA 91 16    JP    M,1691
168D  ED 5B 36 B3 LD    DE,(B336)
1691  2A 36 B3    LD    HL,(B336)
1694  2B          DEC   HL
1695  B7          OR    A
1696  ED 42       SBC   HL,BC
1698  F2 AD 16    JP    P,16AD
169B  2A 34 B3    LD    HL,(B334)
169E  B7          OR    A
169F  ED 42       SBC   HL,BC
16A1  F2 A8 16    JP    P,16A8
16A4  ED 4B 34 B3 LD    BC,(B334)
16A8  EB          EX    DE,HL
```

```
16A9  D1            POP   DE
16AA  37            SCF
16AB  C9            RET
16AC  E1            POP   HL
16AD  D1            POP   DE
16AE  B7            OR    A
16AF  C9            RET

***********************************  Test, ob Koordin. für horizontale
                                     Linie innerhalb Grenzen sind
                                     IN/OUT: DE: X-Startkoordinate
                                             BC: X-Endkoordinate
                                             HL: Y-Koordinate
                                             (reale Koordinaten!)
                                     OUT: CY=0 für außerhalb Grenzen
16B0  E5            PUSH  HL           Y-Koordinate
16B1  D5            PUSH  DE           X-Startkoordinate
16B2  EB            EX    DE,HL        Y-Koordinate nach DE
16B3  2A 36 B3      LD    HL,(B336)    untere Grenze
16B6  2B            DEC   HL
16B7  B7            OR    A
16B8  ED 52         SBC   HL,DE        Y-Kooordinate <= Grenze-1 ?
16BA  F2 F8 16      JP    P,16F8       dann Fehler
16BD  2A 34 B3      LD    HL,(B334)    obere Grenze
16C0  B7            OR    A
16C1  ED 52         SBC   HL,DE        Y-Koordinate > Grenze ?
16C3  FA F8 16      JP    M,16F8       dann Fehler
16C6  D1            POP   DE           X-Startkoordinate
16C7  2A 32 B3      LD    HL,(B332)    rechte Grenze
16CA  B7            OR    A
16CB  ED 52         SBC   HL,DE        X-Start-Koordinate > Grenze ?
16CD  FA F9 16      JP    M,16F9       dann Fehler
16D0  2A 30 B3      LD    HL,(B330)    linke Grenze
16D3  2B            DEC   HL
16D4  B7            OR    A
16D5  ED 52         SBC   HL,DE        X-Start-Koordin. > Grenze-1 ?
16D7  FA DE 16      JP    M,16DE       dann o.k.
16DA  ED 5B 30 B3   LD    DE,(B330)    sonst linke Grenze als X-Start
16DE  2A 30 B3      LD    HL,(B330)    linke Grenze
16E1  2B            DEC   HL
16E2  B7            OR    A
16E3  ED 42         SBC   HL,BC        X-Endkoordinate < Grenze-1 ?
16E5  F2 F9 16      JP    P,16F9       dann Fehler
16E8  2A 32 B3      LD    HL,(B332)    rechte Grenze
16EB  B7            OR    A
16EC  ED 42         SBC   HL,BC        X-Endkoordinate < Grenze ?
16EE  F2 F5 16      JP    P,16F5       dann o.k.
16F1  ED 4B 32 B3   LD    BC,(B332)    sonst rechte Grenze als X-Ende
16F5  E1            POP   HL           Y-Koordinate zurück
16F6  37            SCF                CY=1 für o.k.
16F7  C9            RET
16F8  D1            POP   DE           X-Startkoordinate
16F9  E1            POP   HL           Y-Koordinate
16FA  B7            OR    A            CY=0 für außerhalb Grenzen
16FB  C9            RET
```

```
**********************************   Test, ob Koord. innerhalb Grenzen
                                     IN : DE: X-Koordinate
                                          HL: Y-Koordinate
                                     OUT: DE,HL: reale Koordinaten
                                          CY=0 für außerhalb Grenzen
16FC  CD 1D 16    CALL   161D          Curpos. setzen, reale K. ber.
16FF  E5          PUSH   HL            Y-Koordinate retten
1700  2A 30 B3    LD     HL,(B330)     linke Grenze
1703  2B          DEC    HL
1704  B7          OR     A
1705  ED 52       SBC    HL,DE         X-Koordinate <= Grenze-1 ?
1707  F2 2D 17    JP     P,172D        dann Fehler
170A  2A 32 B3    LD     HL,(B332)     rechte Grenze
170D  B7          OR     A
170E  ED 52       SBC    HL,DE         X-Koordinate > Grenze ?
1710  FA 2D 17    JP     M,172D        dann Fehler
1713  E1          POP    HL            Y-Koordinate
1714  D5          PUSH   DE            X-Koordinate retten
1715  EB          EX     DE,HL         Y-Koordinate nach DE
1716  2A 36 B3    LD     HL,(B336)     untere Grenze
1719  2B          DEC    HL
171A  B7          OR     A
171B  ED 52       SBC    HL,DE         Y-Koordinate <= Grenze-1 ?
171D  F2 30 17    JP     P,1730        dann Fehler
1720  2A 34 B3    LD     HL,(B334)     obere Grenze
1723  B7          OR     A
1724  ED 52       SBC    HL,DE         Y-Koordinate > Grenze ?
1726  FA 30 17    JP     M,1730        dann Fehler
1729  EB          EX     DE,HL         Y-Koordinate nach HL
172A  D1          POP    DE            X-Koordinate zurück
172B  37          SCF                  CY=1 für o.k.
172C  C9          RET
172D  E1          POP    HL            Y-Koordinate
172E  B7          OR     A             CY=0 für außerhalb Grenzen
172F  C9          RET
1730  EB          EX     DE,HL         Y-Koordinate nach HL
1731  D1          POP    DE            X-Koordinate zurück
1732  B7          OR     A             CY=0 für außerhalb Grenzen
1733  C9          RET

**********************************   GRA WIN WIDTH
                                     IN : DE: linke Windowgrenze
                                          HL: rechte Windowgrenze
1734  E5          PUSH   HL
1735  CD 60 17    CALL   1760           linke Grenze in zul. Bereich
1738  D1          POP    DE
1739  E5          PUSH   HL
173A  CD 60 17    CALL   1760          rechte Grenze in zul. Bereich
173D  D1          POP    DE
173E  7B          LD     A,E             linke Grenze < rechte ?
173F  95          SUB    L
1740  7A          LD     A,D
1741  9C          SBC    H
1742  38 01       JR     C,1745        dann o.k.
1744  EB          EX     DE,HL         sonst Grenzen vertauschen
1745  7B          LD     A,E             linke Grenze auf Byteanfang
1746  E6 F8       AND    F8              runden
1748  5F          LD     E,A
```

```
1749  7D           LD     A,L          rechte Grenze auf Byteende
174A  F6 07        OR     07           runden
174C  6F           LD     L,A
174D  CD EC 0A     CALL   0AEC       Mode-Nummer holen
1750  3D           DEC    A          Mode 0 ?
1751  FC 70 17     CALL   M,1770     dann Koordinaten halbieren
1754  3D           DEC    A          Mode 0 oder Mode 1 ?
1755  FC 70 17     CALL   M,1770     dann Koordinaten halbieren
1758  ED 53 30 B3  LD     (B330),DE  Grenze links
175C  22 32 B3     LD     (B332),HL  und rechts setzen
175F  C9           RET

**********************************  X-Grenze in zulässige Grenzen
                                    IN : DE: X-Windowgrenze
                                    OUT: HL: X-Windowgrenze
1760  7A           LD     A,D
1761  B7           OR     A
1762  21 00 00     LD     HL,0000
1765  F8           RET    M          Grenze < 0 ? dann Grenze = 0
1766  21 7F 02     LD     HL,027F    max. Wert = 639
1769  7B           LD     A,E
176A  95           SUB    L
176B  7A           LD     A,D          Grenze >= 639 ?
176C  9C           SBC    H
176D  D0           RET    NC         dann max. Wert 639
176E  EB           EX     DE,HL      sonst alte Grenze
176F  C9           RET

**********************************  Koordinaten halbieren
                                    IN/OUT: DE,HL: Koordinaten
1770  CB 2A        SRA    D
1772  CB 1B        RR     E
1774  CB 2C        SRA    H
1776  CB 1D        RR     L
1778  C9           RET

**********************************  GRA WIN HEIGHT
                                    IN : HL: untere Grenze
                                         DE: obere Grenze
1779  E5           PUSH   HL
177A  CD 92 17     CALL   1792       obere Grenze in zul. Bereich
177D  D1           POP    DE
177E  E5           PUSH   HL
177F  CD 92 17     CALL   1792       untere Grenze in zul. Bereich
1782  D1           POP    DE
1783  7D           LD     A,L
1784  93           SUB    E            untere < obere Grenze ?
1785  7C           LD     A,H
1786  9A           SBC    D
1787  38 01        JR     C,178A     dann o.k.
1789  EB           EX     DE,HL      sonst Grenzen vertauschen
178A  ED 53 34 B3  LD     (B334),DE  obere
178E  22 36 B3     LD     (B336),HL  und untere Grenze setzen
1791  C9           RET
```

```
************************************   Y-Grenze in zulässigen Bereich
                                       IN : DE: Y-Windowgrenze
                                       OUT: HL: Y-Windowgrenze
1792  7A           LD     A,D
1793  B7           OR     A
1794  21 00 00     LD     HL,0000
1797  F8           RET    M              Grenze < 0 ? dann Grenze = 0
1798  CB 3A        SRL    D                Grenze / 2 (reale
179A  CB 1B        RR     E                Koordinate herstellen)
179C  21 C7 00     LD     HL,00C7        max. Wert = 199
179F  7B           LD     A,E
17A0  95           SUB    L                Grenze >= 199 ?
17A1  7A           LD     A,D
17A2  9C           SBC    H
17A3  D0           RET    NC             dann max. Wert 199
17A4  EB           EX     DE,HL          sonst alte Grenze
17A5  C9           RET

************************************   GRA GET WINDOW WIDTH
                                       OUT: DE: linke Windowgrenze
                                            HL: rechte Windowgrenze
17A6  ED 5B 30 B3  LD     DE,(B330)      reale linke
17AA  2A 32 B3     LD     HL,(B332)      und rechte Grenze laden
17AD  CD EC 0A     CALL   0AEC           Mode-Nummer holen
17B0  3D           DEC    A              Mode 0 ?
17B1  FC B6 17     CALL   M,17B6         dann Grenzen verdoppeln
17B4  3D           DEC    A              Mode 2 ?
17B5  F0           RET    P              dann fertig
17B6  29           ADD    HL,HL          rechte Grenze verdoppeln
17B7  23           INC    HL             Grenze auf Byteende
17B8  EB           EX     DE,HL
17B9  29           ADD    HL,HL          linke Grenze verdoppeln
17BA  EB           EX     DE,HL
17BB  C9           RET

************************************   GRA GET WINDOW HEIGHT
                                       OUT: DE: obere Windowgrenze
                                            HL: untere Windowgrenze
17BC  ED 5B 34 B3  LD     DE,(B334)      reale obere
17C0  2A 36 B3     LD     HL,(B336)      und untere Grenze laden
17C3  18 F1        JR     17B6           Grenzen verdoppeln

************************************   GRA CLEAR WINDOW
17C5  CD A6 17     CALL   17A6           Windowgrenzen links/rechts
17C8  B7           OR     A
17C9  ED 52        SBC    HL,DE          rechte - linke Grenze
17CB  23           INC    HL             +1 ergibt Breite
17CC  CD 74 17     CALL   1774             Breite durch 8
17CF  CD 74 17     CALL   1774             teilen, ergibt Zahl der
17D2  CB 3D        SRL    L                Bytes pro Rasterzeile
17D4  45           LD     B,L            nach B
17D5  ED 5B 36 B3  LD     DE,(B336)      reale untere
17D9  2A 34 B3     LD     HL,(B334)      und obere Grenze
17DC  E5           PUSH   HL
17DD  B7           OR     A
17DE  ED 52        SBC    HL,DE          obere - untere Grenze
17E0  23           INC    HL             +1 ergibt Höhe
17E1  4D           LD     C,L            als Zahl der Rasterzeilen
```

```
17E2  ED 5B 30 B3  LD     DE,(B330)     linke Grenze
17E6  E1           POP    HL            obere Grenze
17E7  C5           PUSH   BC            Bytes/Zeile und Rasterzeilenz.
17E8  CD A9 0B     CALL   0BA9          Bildschirmadr. und Maske holen
17EB  D1           POP    DE            Bytes/Zeile und Rasterzeilenz.
17EC  3A 39 B3     LD     A,(B339)      Paper-Byte
17EF  4F           LD     C,A           nach C
17F0  CD B7 0D     CALL   0DB7          Bereich füllen
17F3  C3 0B 16     JP     160B          Cursorposition = 0/0

*********************************     GRA SET PEN
                                      IN : A: Farbstiftnummer
17F6  CD 86 0C     CALL   0C86          Stiftnr. in Farbmaske wandeln
17F9  32 38 B3     LD     (B338),A      und Farbmaske speichern
17FC  C9           RET

*********************************     GRA SET PAPER
                                      IN : A: Farbstiftnummer
17FD  CD 86 0C     CALL   0C86          Stiftnr. in Farbmaske wandeln
1800  32 39 B3     LD     (B339),A      und Farbmaske speichern
1803  C9           RET

*********************************     GRA GET PEN
                                      OUT: A: Farbstiftnummer
1804  3A 38 B3     LD     A,(B338)      Farbmaske
1807  C3 A0 0C     JP     0CA0          in Farbstiftnummer wandeln

*********************************     GRA GET PAPER
                                      OUT: A: Farbstiftnummer
180A  3A 39 B3     LD     A,(B339)      Farbmaske
180D  C3 A0 0C     JP     0CA0          in Farbstiftnummer wandeln

*********************************     GRA PLOT RELATIVE
                                      IN : DE: Cursor-relative X-Koord.
                                           HL: Cursor-relative Y-Koord.
1810  CD 57 16     CALL   1657          relative in absolute Koord.

*********************************     GRA PLOT ABSOLUTE
                                      IN : DE: X-Koordinate
                                           HL: Y-Koordinate
1813  C3 DC BD     JP     BDDC          Indirection zu GRA PLOT

*********************************     GRA PLOT
                                      IN : DE: X-Koordinate
                                           HL: Y-Koordinate
1816  CD FC 16     CALL   16FC          liegt Punkt im Window ?
1819  D0           RET    NC            nein ? dann fertig
181A  CD A9 0B     CALL   0BA9          Bildschirmadr. und Maske holen
181D  3A 38 B3     LD     A,(B338)      Pen-Maske
1820  47           LD     B,A           nach B
1821  C3 E8 BD     JP     BDE8          SCR WRITE, Pixel setzen

*********************************     GRA TEST RELATIVE
                                      IN : DE: Cursor-relative X-Koord.
                                           HL: Cursor-relative Y-Koord.
                                      OUT: A: Farbstiftnr. des Pixels
1824  CD 57 16     CALL   1657          relative in absolute Koord.
```

```
***********************************    GRA TEST ABSOLUTE
                                       IN : DE: X-Koordinate
                                            HL: Y-Koordinate
                                       OUT: A: Farbstiftnr. des Pixels
1827  C3 DF BD     JP     BDDF            Indirection zu GRA TEST

***********************************    GRA TEST
                                       IN : DE: X-Koordinate
                                            HL: Y-Koordinate
                                       OUT: A: Farbstiftnr. des Pixels
182A  CD FC 16     CALL   16FC            liegt Punkt im Window ?
182D  D2 0A 18     JP     NC,180A         nein ? dann Paper-Stift holen
1830  CD A9 0B     CALL   0BA9            Bildschirmadr. und Maske holen
1833  C3 E5 BD     JP     BDE5            SCR READ, Pixel-Farbstiftnr.

***********************************    GRA LINE RELATIVE
                                       IN : DE: Cursor-relative X-Koord.
                                            HL: Cursor-relative Y-Koord.
1836  CD 57 16     CALL   1657            relative in absolute Koord.

***********************************    GRA LINE ABSOLUTE
                                       IN : DE: X-Koordinate
                                            HL: Y-Koordinate
1839  C3 E2 BD     JP     BDE2            Indirection zu GRA LINE

***********************************    GRA LINE
                                       IN : DE: X-Koordinate
                                            HL: Y-Koordinate
183C  E5           PUSH   HL                Y- und X-Koordiante als
183D  D5           PUSH   DE                Endkoordinaten retten
183E  CD 1A 16     CALL   161A            reale Cursorkoordinaten holen
1841  ED 53 42 B3  LD     (B342),DE         und als Startkoordinaten
1845  22 44 B3     LD     (B344),HL         speichern
1848  D1           POP    DE
1849  E1           POP    HL                Endkoordinaten zurück
184A  CD 1D 16     CALL   161D            als Cursorp. setzen, reale K.
184D  E5           PUSH   HL              Y-Endkoordinate retten
184E  2A 42 B3     LD     HL,(B342)       X-Startkoordinate
1851  B7           OR     A
1852  ED 52        SBC    HL,DE           - X-Endkoordinate
1854  44           LD     B,H               gibt X-Differenz, nach BC
1855  4D           LD     C,L
1856  FA 69 18     JP     M,1869          X-Endkoordinate größer ?
1859  2A 42 B3     LD     HL,(B342)         sonst X- und Y-
185C  EB           EX     DE,HL             Start- und -Endkoordinaten
185D  22 42 B3     LD     (B342),HL         vertauschen
1860  2A 44 B3     LD     HL,(B344)
1863  E3           EX     (SP),HL
1864  22 44 B3     LD     (B344),HL
1867  18 08        JR     1871
1869  21 00 00     LD     HL,0000
186C  B7           OR     A                 Zweierkomplement der
186D  ED 42        SBC    HL,BC             negativen X-Differenz
186F  44           LD     B,H               (also Betrag) bilden
1870  4D           LD     C,L
1871  D1           POP    DE              Y-Endkoordinate
1872  2A 44 B3     LD     HL,(B344)       Y-Startkoordinate
1875  B7           OR     A
```

```
1876  ED 52        SBC   HL,DE        Start- minus Endkoordinate
1878  EB           EX    DE,HL        Y-Differenz nach DE
1879  F2 8E 18     JP    P,188E       Y-Differenz positiv ?
187C  21 00 00     LD    HL,0000        sonst Zweierkomplement der
187F  B7           OR    A              negativen Y-Differenz
1880  ED 52        SBC   HL,DE          (also Betrag) bilden
1882  54           LD    D,H            und nach DE
1883  5D           LD    E,L
1884  B7           OR    A
1885  ED 42        SBC   HL,BC        Y-Differenz minus X-Differenz
1887  21 01 00     LD    HL,0001      kleinere Schrittweite =1
188A  30 27        JR    NC,18B3      Y-Differenz >= X-Differenz ?
188C  18 0A        JR    1898
188E  62           LD    H,D
188F  6B           LD    L,E            Y-Differenz nach DE
1890  B7           OR    A
1891  ED 42        SBC   HL,BC        Y-Differenz minus X-Differenz
1893  21 FF FF     LD    HL,FFFF      kleinere Schrittweite =-1
1896  30 09        JR    NC,18A1      Y-Differenz >= X-Differenz ?
1898  22 3A B3     LD    (B33A),HL    kleinere Schrittweite setzen
189B  60           LD    H,B            X-Differenz als größere
189C  69           LD    L,C            Differenz nach HL
189D  3E FF        LD    A,FF         Flag für X-Differenz größer
189F  18 19        JR    18BA
18A1  E5           PUSH  HL           kleinere Schrittweite retten
18A2  2A 42 B3     LD    HL,(B342)      X-Differenz zu (kleinerem)
18A5  09           ADD   HL,BC          X-Start addieren (X-Ende als
18A6  22 42 B3     LD    (B342),HL      neuen Startwert)
18A9  2A 44 B3     LD    HL,(B344)
18AC  B7           OR    A              Y-Differenz von (größerem)
18AD  ED 52        SBC   HL,DE          Y-Start abziehen (Y-Ende als
18AF  22 44 B3     LD    (B344),HL      neuen Startwert
18B2  E1           POP   HL           kleinere Schrittweite
18B3  22 3A B3     LD    (B33A),HL    setzen
18B6  60           LD    H,B            (kleinere) X-Differenz
18B7  69           LD    L,C            nach BC
18B8  EB           EX    DE,HL        Differenzen vertauschen
18B9  AF           XOR   A            Flag für Y-Differenz größer
18BA  32 46 B3     LD    (B346),A     Flag für größere Differenz
18BD  13           INC   DE           kl. Diff. +1 = kleinere Breite
18BE  ED 53 40 B3  LD    (B340),DE    speichern
18C2  23           INC   HL           gr. Diff. +1 = größere Breite
18C3  CD 8C 37     CALL  378C         größere durch kleinere Breite
18C6  22 3C B3     LD    (B33C),HL    Quotient als größere Schrittw.
18C9  ED 53 3E B3  LD    (B33E),DE    Divisionsrest
18CD  ED 4B 40 B3  LD    BC,(B340)    kleinere Breite als Zähler
18D1  50           LD    D,B
18D2  59           LD    E,C            kleinere Breite durch 2
18D3  CB 3A        SRL   D              als Startwert für Summe
18D5  CB 1B        RR    E              der Teilungsreste nach DE
18D7  C5           PUSH  BC           Zähler retten
18D8  ED 4B 3C B3  LD    BC,(B33C)    größere Schrittweite
18DC  2A 3E B3     LD    HL,(B33E)    Rest der gr. Schrittweite
18DF  19           ADD   HL,DE          Rest zu Restsumme
18E0  EB           EX    DE,HL          addieren
18E1  2A 40 B3     LD    HL,(B340)    kleinere Breite (Divisor)
18E4  B7           OR    A
18E5  ED 52        SBC   HL,DE          erreicht Summe der Reste
```

```
18E7  30 07        JR    NC,18F0       noch nicht kleinere Breite ?
18E9  19           ADD   HL,DE       sonst kleinere Breite herst.
18EA  EB           EX    DE,HL
18EB  B7           OR    A             kleinere Breite von Summe
18EC  ED 52        SBC   HL,DE         der Reste abziehen
18EE  EB           EX    DE,HL       Summe der Reste wieder nach DE
18EF  03           INC   BC          gr. Schrittw. zum Ausgl. erh.
18F0  D5           PUSH  DE          Summe der Reste
18F1  3A 46 B3     LD    A,(B346)    Differenzen-Flag
18F4  B7           OR    A
18F5  28 23        JR    Z,191A      Y-Differenz größer ?
18F7  2A 42 B3     LD    HL,(B342)   laufende X-Koordinate
18FA  54           LD    D,H           nach DE
18FB  5D           LD    E,L
18FC  09           ADD   HL,BC       größere (X-)Schrittweite add.
18FD  22 42 B3     LD    (B342),HL   als neue X-Koordinate
1900  44           LD    B,H           nächste X-Startkoordinate
1901  4D           LD    C,L           -1 = aktuelle
1902  0B           DEC   BC            X-Endkoordinate
1903  2A 44 B3     LD    HL,(B344)   laufende Y-Koordinate
1906  E5           PUSH  HL          retten
1907  CD B0 16     CALL  16B0        Koordin. innerhalb Grenzen ?
190A  3A 38 B3     LD    A,(B338)    Pen-Byte
190D  DC C4 0F     CALL  C,0FC4      dann horizontale Linie ziehen
1910  D1           POP   DE          Y-Koordinate
1911  2A 3A B3     LD    HL,(B33A)   kleinere (Y-)Schrittweite
1914  19           ADD   HL,DE       addieren
1915  22 44 B3     LD    (B344),HL   Y-Koordinate neu setzen
1918  18 23        JR    193D
191A  2A 44 B3     LD    HL,(B344)
191D  54           LD    D,H           wenn Y-Differenz größer,
191E  5D           LD    E,L           dann analog vertikale Linie
191F  09           ADD   HL,BC         ziehen
1920  22 44 B3     LD    (B344),HL
1923  44           LD    B,H
1924  4D           LD    C,L
1925  0B           DEC   BC
1926  EB           EX    DE,HL
1927  ED 5B 42 B3  LD    DE,(B342)
192B  D5           PUSH  DE
192C  CD 64 16     CALL  1664
192F  3A 38 B3     LD    A,(B338)
1932  DC 2F 10     CALL  C,102F
1935  D1           POP   DE
1936  2A 3A B3     LD    HL,(B33A)
1939  19           ADD   HL,DE
193A  22 42 B3     LD    (B342),HL
193D  D1           POP   DE          Summe der Reste
193E  C1           POP   BC          Zähler
193F  0B           DEC   BC
1940  78           LD    A,B
1941  B1           OR    C           weitere Durchläufe ?
1942  20 93        JR    NZ,18D7
1944  C9           RET

**********************************  GRA WR CHAR
                                    IN : A: Zeichen

1945  DD E5        PUSH  IX
```

```
1947  CD D3 12     CALL   12D3      Adresse der Zeichenmatrix hol.
194A  11 3A B3     LD     DE,B33A   Adresse für Matrix-Zwischensp.
194D  D5           PUSH   DE
194E  DD E1        POP    IX          nach IX
1950  01 08 00     LD     BC,0008   Länge der Matrix
1953  ED B0        LDIR             Zeichenmatrix kopieren
1955  CD 1A 16     CALL   161A      reale Cursorposition holen
1958  CD FF 16     CALL   16FF      liegt zugeh. Punkt im Window ?
195B  30 4C        JR     NC,19A9   nein ?
195D  E5           PUSH   HL          Cursorposition
195E  D5           PUSH   DE          retten
195F  01 07 00     LD     BC,0007
1962  EB           EX     DE,HL       linke X-Koordinate des
1963  09           ADD    HL,BC       Zeichens = rechte Koord.
1964  EB           EX     DE,HL
1965  B7           OR     A           obere Y-Koordinate -7
1966  ED 42        SBC    HL,BC       =untere Y-Koordinate
1968  CD FF 16     CALL   16FF      liegt zugeh. Punkt im Window ?
196B  D1           POP    DE          Cursorpos. (Zeichen oben/
196C  E1           POP    HL          links) wieder zurück
196D  30 3A        JR     NC,19A9   Pos. unten/rechts außerh. W. ?
196F  CD A9 0B     CALL   0BA9      Bildschirmadr. und Maske holen
1972  16 08        LD     D,08      8 Rasterzeilen
1974  E5           PUSH   HL        Bildschirmadresse retten
1975  1E 08        LD     E,08      8 Spalten
1977  CD CF 19     CALL   19CF      Pixel setzen
197A  CB 09        RRC    C         Maske für nächstes Pixel
197C  DC F9 0B     CALL   C,0BF9    ggf. Adr. des nächsten Bytes
197F  DD CB 00 06  RLC    (IX+00)   näch. Pixel in gepackter Matr.
1983  1D           DEC    E
1984  20 F1        JR     NZ,1977   weitere Spalten ?
1986  E1           POP    HL        Bildschirmadr. der 1. Spalte
1987  CD 13 0C     CALL   0C13      Adr. d. nächsten Rasterzeile
198A  DD 23        INC    IX        nächstes Byte in Matrix
198C  15           DEC    D
198D  20 E5        JR     NZ,1974   weitere Rasterzeilen ?
198F  DD E1        POP    IX        alter Wert (bei Aufruf)
1991  CD FC 15     CALL   15FC      Cursorposition holen
1994  EB           EX     DE,HL     X-Koordinate nach HL
1995  CD EC 0A     CALL   0AEC      Mode-Nummer holen
1998  01 08 00     LD     BC,0008   8 Positionen pro Zeichen
199B  FE 01        CP     01        Mode 1 ?
199D  28 04        JR     Z,19A3    dann 16 Positionen
199F  30 03        JR     NC,19A4   Mode 0 ? dann 8 Positionen
19A1  09           ADD    HL,BC       sonst 32 Positionen
19A2  09           ADD    HL,BC       für Mode 0
19A3  09           ADD    HL,BC
19A4  09           ADD    HL,BC       zu X-Koordinate addieren
19A5  EB           EX     DE,HL     X-Koordinate nach DE
19A6  C3 F4 15     JP     15F4      Cursorposition neu setzen

19A9  0E 08        LD     C,08      8 Rasterzeilen
19AB  D5           PUSH   DE        X-Koordinate
19AC  06 08        LD     B,08      8 Spalten
19AE  CD FF 16     CALL   16FF      liegt Punkt im Window ?
19B1  30 0C        JR     NC,19BF   nein ? dann nicht setzen
19B3  E5           PUSH   HL
19B4  D5           PUSH   DE
```

```
19B5  C5           PUSH  BC
19B6  CD A9 0B     CALL  0BA9          Bildschirmadr. und Maske holen
19B9  CD CF 19     CALL  19CF          Pixel setzen
19BC  C1           POP   BC
19BD  D1           POP   DE
19BE  E1           POP   HL
19BF  DD CB 00 06  RLC   (IX+00)       nächstes Zeichenmatrix-Bit
19C3  13           INC   DE            X-Koordinate erhöhen
19C4  10 E8        DJNZ  19AE          weitere Spalten ?
19C6  D1           POP   DE            X-Koordinate f. 1. Spalte
19C7  2B           DEC   HL            Y-Koordinate herunterzählen
19C8  DD 23        INC   IX            nächstes Matrixbyte
19CA  0D           DEC   C
19CB  20 DE        JR    NZ,19AB       weitere Rasterzeilen ?
19CD  18 C0        JR    198F          Cursor neu setzen

************************************  Pixel für Buchstaben setzen
                                      IN : HL: Bildschirmadresse
                                           IX: Zeichenmatrixadresse
                                           C: Maske f. Pixelauswahl
19CF  DD CB 00 7E  BIT   7,(IX+00)     Bit aus Matrix testen
19D3  3A 38 B3     LD    A,(B338)      Pen-Maske
19D6  20 03        JR    NZ,19DB       Bit gesetzt ?
19D8  3A 39 B3     LD    A,(B339)      sonst Paper-Maske
19DB  47           LD    B,A           Farbmaske nach B
19DC  C3 E8 BD     JP    BDE8          SCR WRITE, Pixel setzen

19DF  C7           RST   00


---------------------- KEYBOARD MANAGER (KM) ----------------------

************************************  KM INITIALIZE
19E0  21 02 1E     LD    HL,1E02       Default Verzögerung
19E3  CD 6D 1C     CALL  1C6D          setzen
19E6  AF           XOR   A
19E7  32 0B B5     LD    (B50B),A      Spaltencode f. höchste Taste
19EA  67           LD    H,A           Shift Lock State
19EB  6F           LD    L,A           und Caps Lock State
19EC  22 E7 B4     LD    (B4E7),HL     löschen
19EF  21 3C B4     LD    HL,B43C       Adr. Repeat-Tabelle
19F2  11 B0 FF     LD    DE,FFB0       -$50=-80, Länge einer Tabelle
19F5  22 47 B5     LD    (B547),HL     Adr. Repeat-Tabelle setzen
19F8  19           ADD   HL,DE         -80=Adr. Key Ctrl Table
19F9  22 45 B5     LD    (B545),HL     setzen
19FC  19           ADD   HL,DE         -80=Adr. Key Shift Table
19FD  22 43 B5     LD    (B543),HL     setzen
1A00  19           ADD   HL,DE         -80=Adr. Key Translation Table
1A01  22 41 B5     LD    (B541),HL     setzen
1A04  EB           EX    DE,HL         und nach DE
1A05  21 69 1D     LD    HL,1D69       Beginn der Default-Tab. im Rom
1A08  01 FA 00     LD    BC,00FA       Länge der Default-Werte
1A0B  ED B0        LDIR                Tabellen kopieren
1A0D  06 0A        LD    B,0A          Länge der Key State Map
1A0F  21 EB B4     LD    HL,B4EB       Adr. der Key State Map
1A12  36 00        LD    (HL),00       Byte der KSM löschen
1A14  23           INC   HL            Zeiger auf nächstes Byte
1A15  10 FB        DJNZ  1A12          bis zum Ende der KSM
```

```
1A17  06 0A       LD     B,0A          Länge der Feedback-Tabelle
1A19  36 FF       LD     (HL),FF       Feedback-Byte auf $FF setzen
1A1B  23          INC    HL            Zeiger auf nächstes Byte
1A1C  10 FB       DJNZ   1A19          bis Ende d. Feedback-Tabelle

*********************************** KM RESET
1A1E  CD ED 1C    CALL   1CED          Ringbuffer initialisieren
1A21  CD 75 1A    CALL   1A75          Put Back Buffer löschen
1A24  11 46 B4    LD     DE,B446       Exp Buffer Start
1A27  21 98 00    LD     HL,0098       Länge des Exp Buffer
1A2A  CD 81 1A    CALL   1A81          Exp Buffer initialisieren
1A2D  21 36 1A    LD     HL,1A36       Start der Indirections
1A30  CD 8A 0A    CALL   0A8A          Indirections kopieren
1A33  C3 82 1C    JP     1C82          Break Event ausschalten

*********************************** Indirection f. KM TEST BREAK
1A36  03                               Anz. d. zu kopierenden Bytes
1A37  EE BD                            Zieladresse im Ram
1A39  C3 2F 1C    JP     1C2F          KM TST BREAK

*********************************** KM WAIT CHAR
                                    OUT: A: Zeichen
1A3C  CD 42 1A    CALL   1A42          Zeichen holen
1A3F  30 FB       JR     NC,1A3C       bis gültiges Zeichen kommt
1A41  C9          RET

*********************************** KM READ CHAR
                                    OUT: A: Zeichen
                                           CY:=1, wenn Zeichen gültig
1A42  E5          PUSH   HL
1A43  21 E0 B4    LD     HL,B4E0       Zeiger Put Back Buffer
1A46  7E          LD     A,(HL)        Zeichen daraus laden
1A47  36 FF       LD     (HL),FF       Put Back Buffer löschen
1A49  BE          CP     (HL)          war Buffer leer?
1A4A  38 27       JR     C,1A73        sonst o.k., raus
1A4C  2A DE B4    LD     HL,(B4DE)     Exp String Code u. Count laden
1A4F  7C          LD     A,H           Exp String Code
1A50  B7          OR     A             <>0?
1A51  20 11       JR     NZ,1A64       dann Zeichen aus Exp String holen
1A53  CD 5C 1B    CALL   1B5C          Zeichen von Tastaturabfrage holen
1A56  30 1B       JR     NC,1A73       kein Zeichen? dann raus
1A58  FE 80       CP     80            ASCII-Zeichen?
1A5A  38 17       JR     C,1A73        dann raus, Zeichen in A
1A5C  FE A0       CP     A0            >$A0?
1A5E  3F          CCF
1A5F  38 12       JR     C,1A73        dann raus, Zeichen in A
1A61  67          LD     H,A           als Exp Code nach H
1A62  2E 00       LD     L,00          Exp String Count :=0 setzen
1A64  D5          PUSH   DE
1A65  CD 2E 1B    CALL   1B2E          Zeichen aus Exp String holen
1A68  38 02       JR     C,1A6C        Exp String bereits zuende?
1A6A  26 00       LD     H,00          dann Code:=0, Exp String aussch.
1A6C  2C          INC    L             Zähler erhöhen
1A6D  22 DE B4    LD     (B4DE),HL     Code und Zähler abspeichern
1A70  D1          POP    DE
1A71  30 E0       JR     NC,1A53       bei Stringende Taste holen
1A73  E1          POP    HL
1A74  C9          RET
```

```
***********************************   Put Back Buffer löschen
1A75  3E FF        LD     A,FF           $FF als Kennzeichen f. PBB leer

***********************************   KM CHAR RETURN
                                      IN : A: Zeichen
1A77  32 E0 B4     LD     (B4E0),A       Zeichen in Put Back Buffer
1A7A  C9           RET

***********************************   KM EXP BUFFER RESET
                                      IN : DE: Start d. Exp Buffers
                                           HL: Länge des Exp Buffers
                                      OUT: CY:=1, wenn o.k.
1A7B  CD 81 1A     CALL   1A81           Exp Buffer initialisieren
1A7E  3F           CCF                   Carry invertieren
1A7F  FB           EI
1A80  C9           RET

***********************************   KM EXP BUFFER CONT'D
                                      IN : DE: Start Expansion Buffer
                                           HL: Länge Expansion Buffer
1A81  F3           DI
1A82  7D           LD     A,L              Länge-31 (Anz. Exp Strings)
1A83  D6 31        SUB    31               d.h. auf Mindestgröße
1A85  7C           LD     A,H              testen
1A86  DE 00        SBC    00
1A88  D8           RET    C              Exp Buffer zu klein? dann raus
1A89  19           ADD    HL,DE          Start+Länge=Ende, nach HL
1A8A  22 E3 B4     LD     (B4E3),HL      Ende Exp Buffer setzen
1A8D  EB           EX     DE,HL
1A8E  22 E1 B4     LD     (B4E1),HL      Anfang Exp Buffer setzen
1A91  01 30 0A     LD     BC,0A30          Default Werte
1A94  36 01        LD     (HL),01          für die Ziffern
1A96  23           INC    HL               0-9 des Zehnerblockes setzen
1A97  71           LD     (HL),C           ( 01 30 01 31 01 32 ... )
1A98  23           INC    HL
1A99  0C           INC    C
1A9A  10 F8        DJNZ   1A94
1A9C  EB           EX     DE,HL            Tabelle d. Defaults f.
1A9D  21 B3 1A     LD     HL,1AB3          ".", ENTER u. CTRL-ENTER
1AA0  0E 0A        LD     C,0A             des Zehnerblockes
1AA2  ED B0        LDIR                    kopieren
1AA4  EB           EX     DE,HL
1AA5  06 13        LD     B,13             restliche
1AA7  AF           XOR    A                19 Exp Strings
1AA8  77           LD     (HL),A           als Leer-Strings
1AA9  23           INC    HL               (Länge=0)
1AAA  10 FC        DJNZ   1AA8             setzen
1AAC  22 E5 B4     LD     (B4E5),HL      Zeiger auf freien Exp Buffer
1AAF  32 DF B4     LD     (B4DF),A       Exp String ausschalten
1AB2  C9           RET

***********************************   Default Expansion Strings

1AB3  01 2E                              .
1AB5  01 0D                              <carriage return>
1AB7  05 52 55 4E 22 0D                  RUN" <carriage return>
```

```
************************************    KM SET EXPAND
                                        IN : B: Code f. Exp String
                                             HL: Zeiger auf den String
                                             C: Länge des Strings
                                        OUT: CY:=1, wenn o.k.
                                               Z:=1, f. Code lfd. String
                                                     = Code neuer String
                                                     ( d. Code in A )
1ABD  F3           DI
1ABE  78           LD     A,B             Code d. Exp Strings
1ABF  CD 3E 1B     CALL   1B3E            lfd. Adresse u. Länge holen
1AC2  30 1F        JR     NC,1AE3         Code illegal? d. CY:=0, raus
1AC4  C5           PUSH   BC              Code in B, Länge in C
1AC5  D5           PUSH   DE              Adr. d. lfd. Exp Strings
1AC6  E5           PUSH   HL              Adr. d. neuen Exp Strings
1AC7  CD E5 1A     CALL   1AE5            Platz f. neuen String schaffen
1ACA  3F           CCF                    CY:=1, wenn o.k.
1ACB  E1           POP    HL
1ACC  D1           POP    DE
1ACD  C1           POP    BC
1ACE  30 13        JR     NC,1AE3         Fehler? dann raus, CY:=0
1AD0  1B           DEC    DE              Zeiger auf Stringlänge
1AD1  79           LD     A,C             Länge d. neuen Strings
1AD2  0C           INC    C               +1 f. Längenbyte vor String
1AD3  12           LD     (DE),A          Zeichen in Exp Buffer
1AD4  13           INC    DE              Zeiger auf nächstes Zeichen
1AD5  E7           RST    20              LD A,(HL) im ganzen Ram
1AD6  23           INC    HL              Zeiger auf nächstes Zeichen
1AD7  0D           DEC    C               Zeichenzähler erniedrigen
1AD8  20 F9        JR     NZ,1AD3         bis zum letzten Zeichen
1ADA  21 DF B4     LD     HL,B4DF         Zeiger Code lfd. Exp String
1ADD  78           LD     A,B             Code f. neuen String
1ADE  AE           XOR    (HL)            m. lfd. Exp String Code verkn.
1ADF  20 01        JR     NZ,1AE2         ungleich? dann o.k., raus
1AE1  77           LD     (HL),A          sonst Code lfd. Exp String
1AE2  37           SCF                    CY:=1 f. o.k.
1AE3  FB           EI
1AE4  C9           RET

************************************    Platz f. neuen String schaffen
                                        IN : A: Länge des alten Strings
                                             C: Länge des neuen Strings
                                             DE: Adr. d. Exp Strings
                                        OUT: CY:=0, wenn o.k.
1AE5  06 00        LD     B,00
1AE7  60           LD     H,B               Länge des alten
1AE8  6F           LD     L,A               Exp Strings
1AE9  79           LD     A,C             - Länge des neuen Strings
1AEA  95           SUB    L               = Offset f. Verschiebung
1AEB  C8           RET    Z               Längen gleich? dann o.k., raus
1AEC  30 0F        JR     NC,1AFD         alter String länger?
1AEE  7D           LD     A,L               Länge d. alten Exp Strings
1AEF  69           LD     L,C               und die des neuen Strings
1AF0  4F           LD     C,A               vertauschen
1AF1  19           ADD    HL,DE           neuer Zeiger nächster String
1AF2  EB           EX     DE,HL           nach DE
1AF3  09           ADD    HL,BC           Zeiger auf nächsten String
1AF4  CD 22 1B     CALL   1B22            Anz. zu versch. Bytes nach BC
```

```
1AF7  28 23       JR    Z,1B1C        keine mehr? d. raus
1AF9  ED B0       LDIR                alle höheren Strings versch.
1AFB  18 1F       JR    1B1C          und raus
1AFD  4F          LD    C,A           Offset
1AFE  19          ADD   HL,DE         Zeiger auf nächsten String
1AFF  E5          PUSH  HL            retten
1B00  2A E5 B4    LD    HL,(B4E5)     Ende der Exp Srings
1B03  09          ADD   HL,BC         +Offset=neues Exp String Ende
1B04  EB          EX    DE,HL         nach DE
1B05  2A E3 B4    LD    HL,(B4E3)     Bufferende
1B08  7D          LD    A,L             neues Stringende
1B09  93          SUB   E               über Bufferende
1B0A  7C          LD    A,H             hinaus?
1B0B  9A          SBC   D
1B0C  E1          POP   HL
1B0D  D8          RET   C             dann CY:=1, Fehler, raus
1B0E  CD 22 1B    CALL  1B22          Anz. zu versch. Bytes nach BC
1B11  2A E5 B4    LD    HL,(B4E5)     Ende der Strings als Quelle
1B14  28 06       JR    Z,1B1C        keine Verschiebung? d. raus
1B16  D5          PUSH  DE            neues Ende der Exp Strings
1B17  1B          DEC   DE              Zeiger auf letztes benutztes
1B18  2B          DEC   HL              Byte setzen
1B19  ED B8       LDDR                und String nach oben kopieren
1B1B  D1          POP   DE            neues Ende der Exp Strings
1B1C  ED 53 E5 B4 LD    (B4E5),DE     setzen
1B20  B7          OR    A             CY:=0 f. o.k.
1B21  C9          RET

*********************************** Anz. zu versch. Bytes berechnen
                                    IN : HL: neue Adr. nächster Str.
                                    OUT: BC: Anz. zu versch. Bytes
1B22  3A E5 B4    LD    A,(B4E5)        Ende der Exp Strings (+1)
1B25  95          SUB   L               -
1B26  4F          LD    C,A             Start des nächstes Strings
1B27  3A E6 B4    LD    A,(B4E6)        =
1B2A  9C          SBC   H               Anz. zu versch. Bytes
1B2B  47          LD    B,A             nach BC
1B2C  B1          OR    C
1B2D  C9          RET

*********************************** KM GET EXPAND
                                    IN : A: Code ($00..$1F,$80..$9F)
                                         L: Exp String Zähler
                                    OUT: A: Zeichen aus Exp String
                                         CY:=1, wenn o.k.; Z:=1, wenn Ende erreicht
1B2E  CD 3E 1B    CALL  1B3E          Adr. d. Exp Strings holen
1B31  D0          RET   NC            Code illegal? d. Fehler, raus
1B32  BD          CP    L             Länge u. lfd. Zähler vergl.
1B33  C8          RET   Z             gleich? dann raus
1B34  3F          CCF                 wenn Fehler, CY:=0
1B35  D0          RET   NC            und raus
1B36  E5          PUSH  HL            Exp String Zähler retten
1B37  26 00       LD    H,00          Hi Byte auf Null setzen
1B39  19          ADD   HL,DE         =Adr. d. gesuchten Zeichens
1B3A  7E          LD    A,(HL)        Zeichen aus Exp String laden
1B3B  E1          POP   HL            Exp String Count zurück
1B3C  37          SCF                 CY:=1 f. o.k.
1B3D  C9          RET
```

```
************************************   Adr. eines Exp Strings berechnen
                                       IN : A: Code ($00..$1F,$80..$9F)
                                       OUT: DE: Adr. d. Exp Strings
                                             A: Länge des Exp Strings
                                             CY:=1, wenn o.k.
1B3E  E6 7F      AND    7F               oberstes Bit :=0 setzen
1B40  FE 20      CP     20               >$1F bzw. >$9F?
1B42  D0         RET    NC               dann Fehler, raus
1B43  E5         PUSH   HL
1B44  2A E1 B4   LD     HL,(B4E1)        Adr. Exp Buffer
1B47  11 00 00   LD     DE,0000
1B4A  3C         INC    A                Stringno. z. Ausgleich erhöhen
1B4B  19         ADD    HL,DE            Länge d. letzten Strings add.
1B4C  5E         LD     E,(HL)           Länge des neuen Strings laden
1B4D  23         INC    HL               Zeiger auf erstes Zeichen
1B4E  3D         DEC    A                Nummer d. Strings erniedrigen
1B4F  20 FA      JR     NZ,1B4B          bis zum gesuchten String
1B51  7B         LD     A,E              Länge d. gesuchten Strings
1B52  EB         EX     DE,HL            Adr. desselben nach DE
1B53  E1         POP    HL
1B54  37         SCF                     CY:=1 f. o.k.
1B55  C9         RET

************************************   KM WAIT KEY
                                       OUT: A: Zeichen von Tastatur
1B56  CD 5C 1B   CALL   1B5C             Zeichen von Tastatur holen
1B59  30 FB      JR     NC,1B56          bis Taste gedrückt
1B5B  C9         RET

************************************   KM READ KEY
                                       OUT: A: Zeichen von Tastatur
                                             CY:=1, wenn Taste gedrückt
1B5C  E5         PUSH   HL
1B5D  C5         PUSH   BC
1B5E  CD 15 1D   CALL   1D15             Tastenkoordinaten aus Ringbuf.
1B61  30 3A      JR     NC,1B9D          Ringbuffer leer? dann raus
1B63  79         LD     A,C              Zeilennummer u. CTRL/SHIFT
1B64  FE EF      CP     EF               BRK?
1B66  28 34      JR     Z,1B9C           dann Code direkt übernehmen
1B68  E6 0F      AND    0F               Zeilennummer
1B6A  87         ADD    A                  mit 8 (f. 8 Tasten pro
1B6B  87         ADD    A                  Zeile) multiplizieren
1B6C  87         ADD    A
1B6D  3D         DEC    A                zum Ausgleich -1
1B6E  3C         INC    A                pro Bitstelle um 1 erhöhen
1B6F  CB 08      RRC    B                Bit ins Carry
1B71  30 FB      JR     NC,1B6E          bis 1-Bit gefunden
1B73  CD A0 1B   CALL   1BA0             Tastennummer in ASCII-Code
1B76  21 E8 B4   LD     HL,B4E8          Caps Lock Flag
1B79  CB 7E      BIT    7,(HL)           gesetzt?
1B7B  28 0A      JR     Z,1B87           sonst nicht wandeln
1B7D  FE 61      CP     61               Zeichen < "a"?
1B7F  38 06      JR     C,1B87           dann raus
1B81  FE 7B      CP     7B               Zeichen > "z"?
1B83  30 02      JR     NC,1B87          dann raus
1B85  C6 E0      ADD    E0               auf Großschrift forcieren
1B87  FE FF      CP     FF               Taste ungültig?
1B89  28 D3      JR     Z,1B5E           d. nächste Taste aus Ringbuf.
```

```
1B8B  FE FE       CP     FE          CTRL-Caps Lock?
1B8D  21 E7 B4    LD     HL,B4E7     d. Shift Lock Flag
1B90  28 05       JR     Z,1B97      umschalten
1B92  FE FD       CP     FD          Caps Lock?
1B94  23          INC    HL          dann Caps Lock Flag
1B95  20 05       JR     NZ,1B9C     umschalten, andernfalls raus
1B97  7E          LD     A,(HL)      entspr. Flag
1B98  2F          CPL                invertieren
1B99  77          LD     (HL),A      und neu setzen
1B9A  18 C2       JR     1B5E        und nächstes Zeichen aus Buf.
1B9C  37          SCF                CY:=1 f. o.k.
1B9D  C1          POP    BC
1B9E  E1          POP    HL
1B9F  C9          RET

***********************************  Keyno.->ASCII wandeln
                                     IN : A: Tastennummer
                                          C: <b7>: CTRL-Flag
                                             <b5>: SHIFT-Flag
                                     OUT: A: ASCII-Code
1BA0  CB 11       RL     C              CTRL-Flag ins Carry
1BA2  DA 48 1D    JP     C,1D48         gesetzt? d. aus CTRL-Tabelle
1BA5  47          LD     B,A            Tastennummer nach B retten
1BA6  3A E7 B4    LD     A,(B4E7)       SHIFT-Lock Flag
1BA9  B1          OR     C              mit SHIFT-Flag verknüpfen
1BAA  E6 40       AND    40             und entspr. Bit isolieren
1BAC  78          LD     A,B            Zeichen zurück
1BAD  C2 43 1D    JP     NZ,1D43        SHIFT? d. aus SHIFT-Tabelle
1BB0  C3 3E 1D    JP     1D3E           sonst einfachen Code holen

***********************************  KM GET STATE
                                     OUT: H: Caps Lock Flag
                                          L: Shift Lock Flag
1BB3  2A E7 B4    LD     HL,(B4E7)      Flags laden
1BB6  C9          RET

***********************************  Update Key State Map
1BB7  11 FF B4    LD     DE,B4FF        Beginn d. pos. Rückmeldungen
1BBA  21 F5 B4    LD     HL,B4F5        Beginn Key Response Table
1BBD  CD 46 08    CALL   0846           Tastatur abfragen
1BC0  3A 01 B5    LD     A,(B501)       Rückmeldung d. Zeile 2
1BC3  E6 A0       AND    A0             CTRL u. SHIFT isolieren
1BC5  4F          LD     C,A            und in C sichern
1BC6  21 ED B4    LD     HL,B4ED        CTRL/SHIFT-Flag
1BC9  B6          OR     (HL)           in alte Rücmeldungen speichern
1BCA  77          LD     (HL),A         um Tastenerkennung zu verhind.
1BCB  21 FF B4    LD     HL,B4FF        Beginn der positiven Rückmeld.
1BCE  11 EB B4    LD     DE,B4EB        Key State Map (KSM)
1BD1  06 00       LD     B,00           Flag f. neue Tasten löschen
1BD3  1A          LD     A,(DE)         Byte aus KSM
1BD4  AE          XOR    (HL)             alte Tasten m. neuen verkn.
1BD5  A6          AND    (HL)             ergibt neu gedr. Tasten
1BD6  C4 48 1C    CALL   NZ,1C48        neue Tasten? d. Tasten in Buf.
1BD9  7E          LD     A,(HL)         lfd. Tasten
1BDA  12          LD     (DE),A         in Key State Map schreiben
1BDB  23          INC    HL             Zeiger auf nächste Rückmeldung
1BDC  13          INC    DE             Zeiger auf nächstes KSM-Byte
1BDD  0C          INC    C              Zeilenzähler erhöhen
```

```
1BDE  79        LD    A,C           und
1BDF  E6 0F     AND   0F            Zeilennummer isolieren
1BE1  FE 0A     CP    0A            bis alle 10 Rückmeldungen
1BE3  20 EE     JR    NZ,1BD3       bearbeitet
1BE5  79        LD    A,C           CTRL- u. SHIFT-Flag
1BE6  E6 A0     AND   A0            isolieren
1BE8  CB 71     BIT   6,C           Taste in Zeile 8? (außer TAB)
1BEA  4F        LD    C,A           isolierte CTRL/SHIFT-Flags
1BEB  C4 EE BD  CALL  NZ,BDEE       ggf. KM TEST BREAK
1BEE  78        LD    A,B           Flag f. neue Tasten
1BEF  B7        OR    A             gab es neue Tasten?
1BF0  C0        RET   NZ            dann zurück
1BF1  21 09 B5  LD    HL,B509       sonst Verzögerungszähler
1BF4  35        DEC   (HL)          erniedrigen
1BF5  C0        RET   NZ            und bei <>0 zurück
1BF6  2A 0A B5  LD    HL,(B50A)     sonst höchste gedrückte Taste
1BF9  EB        EX    DE,HL         nach DE
1BFA  42        LD    B,D           isolierte Rückmeldung nach B
1BFB  16 00     LD    D,00          DE: Zeilennummer, 16-Bit
1BFD  21 EB B4  LD    HL,B4EB       Adr. Key State Map
1C00  19        ADD   HL,DE         ergibt Adr d. Zeilenrückmeld.
1C01  7E        LD    A,(HL)        Rückmeld. Zeile höchste Taste
1C02  2A 47 B5  LD    HL,(B547)     Adr. d. Repeat-Tabelle
1C05  19        ADD   HL,DE         Adr. Repeat-Byte betr. Zeile
1C06  A6        AND   (HL)          Rpt-Byte u. Rückmeldung verkn.
1C07  A0        AND   B             und m. höchster Taste verkn.
1C08  C8        RET   Z             kein Repeat? dann zurück
1C09  21 09 B5  LD    HL,B509       sonst Zähler
1C0C  34        INC   (HL)          wieder erhöhen
1C0D  3A 40 B5  LD    A,(B540)      Anz. d. Tasten im Buffer
1C10  B7        OR    A             wenn Tasten im Buffer,
1C11  C0        RET   NZ            dann zurück
1C12  79        LD    A,C           CTRL/SHIFT-Flags
1C13  B3        OR    E             und Zeilennummer
1C14  4F        LD    C,A           nach C
1C15  3A E9 B4  LD    A,(B4E9)      Reload Count f. Rpt-Verzöger.

***********************************   Tastenkoordinaten in Buffer
                                    IN : A: Repeat-Verzögerung
                                         B: Rückmeldung (ein 1-Bit)
                                         C: Zeilennummer
1C18  32 09 B5  LD    (B509),A      Repeat-Verzögerung setzen
1C1B  CD FE 1C  CALL  1CFE          Tastenkoordinaten in Buffer
1C1E  79        LD    A,C           Zeilennummer
1C1F  E6 0F     AND   0F            isolieren
1C21  6F        LD    L,A           Zeilennummer
1C22  60        LD    H,B           und Rückmeldung
1C23  22 0A B5  LD    (B50A),HL     f. höchste Taste setzen
1C26  FE 08     CP    08            Zeile 8?
1C28  C0        RET   NZ            sonst raus
1C29  CB 60     BIT   4,B           TAB?
1C2B  C0        RET   NZ            dann raus
1C2C  CB F1     SET   6,C           sonst Flag f. Zeile 8 ohne TAB
1C2E  C9        RET
```

```
***********************************      KM TEST BREAK
                                         IN : C: CTRL/SHIFT-Flag
1C2F  21 F3 B4     LD     HL,B4F3          Rückm. Zeile 8
1C32  CB 56        BIT    2,(HL)           ESC gedrückt?
1C34  C8           RET    Z                sonst zurück
1C35  79           LD     A,C              CTRL/SHIFT-Flags
1C36  EE A0        XOR    A0               invertieren
1C38  20 56        JR     NZ,1C90          beide gesetzt? sonst raus
1C3A  C5           PUSH   BC
1C3B  23           INC    HL               Zeiger auf Rückm. Zeile 9
1C3C  06 0A        LD     B,0A             Zähler f. 10 Rückmeldungen
1C3E  8E           ADC    (HL)             Rückmeldungen aufsummieren
1C3F  2B           DEC    HL               Zeiger auf vorhergeh. Rückm.
1C40  10 FC        DJNZ   1C3E             bis alle Rückm. addiert
1C42  C1           POP    BC
1C43  FE A4        CP     A4               Summe=$A0+$04?
1C45  20 49        JR     NZ,1C90          sonst Break Event ausführen
1C47  C7           RST    00               bei CTRL-SHIFT-ESC: Reset

***********************************      Tasten einer Zeile in Buffer
                                         IN : A: Rückmeldung einer Zeile
                                              C: Zeilennummer
1C48  E5           PUSH   HL                  Tabellenzeiger
1C49  D5           PUSH   DE                  retten
1C4A  5F           LD     E,A              Rückmeldung retten
1C4B  2F           CPL                        Zweierkomplement
1C4C  3C           INC    A                   bilden
1C4D  A3           AND    E                unterstes 1-Bit isolieren
1C4E  47           LD     B,A              und nach B
1C4F  3A EA B4     LD     A,(B4EA)         Reload Count f. Anfangsverz.
1C52  CD 18 1C     CALL   1C18             Tastenkoordinaten in Ringbuf.
1C55  78           LD     A,B              1-Bit f. letzte Taste
1C56  AB           XOR    E                aus Zeilenrückm. entfernen
1C57  20 F1        JR     NZ,1C4A          noch Tast. in Zeile? d. bearb.
1C59  D1           POP    DE
1C5A  E1           POP    HL
1C5B  C9           RET

***********************************      KM GET JOYSTICK
                                         OUT: H: Joystick 0
                                              L: Joystick 1
1C5C  3A F1 B4     LD     A,(B4F1)         Rückm. f. Joystick 1
1C5F  E6 7F        AND    7F               b7 eliminieren
1C61  6F           LD     L,A              und nach L
1C62  3A F4 B4     LD     A,(B4F4)         Rückm. f. Joystick 0
1C65  E6 7F        AND    7F               b7 eliminieren
1C67  67           LD     H,A              und nach H
1C68  C9           RET

***********************************      KM GET DELAY
                                         OUT: H: 1. Verzögerung
                                              L: Repeat-Verzögerung
1C69  2A E9 B4     LD     HL,(B4E9)        Verzögerungs-Werte laden
1C6C  C9           RET

***********************************      KM SET DELAY
                                         IN : H: 1. Verzögerung
                                              L: Repeat-Verzögerung
```

```
1C6D  22 E9 B4    LD      (B4E9),HL     Verzögerungswerte setzen
1C70  C9          RET

**********************************    KM ARM BREAK
                                      IN : DE: Routinenadresse
                                            C: Rom-Konfiguration
                                      OUT: HL: Adr. d. User-Bereiches
1C71  CD 82 1C    CALL    1C82             Break Event ausschalten
1C74  21 0D B5    LD      HL,B50D          Start d. Break Event Blocks
1C77  06 40       LD      B,40             Express, Synchronous, Far Call
1C79  CD D2 01    CALL    01D2             Event Block aufbauen
1C7C  3E FF       LD      A,FF             Flag f. Event Block gültig
1C7E  32 0C B5    LD      (B50C),A         setzen
1C81  C9          RET

**********************************    KM DISARM BREAK
1C82  C5          PUSH    BC
1C83  D5          PUSH    DE
1C84  21 0C B5    LD      HL,B50C          Flag f. Break Event inaktiv
1C87  36 00       LD      (HL),00          setzen
1C89  23          INC     HL               Zeiger auf Event Block
1C8A  CD 85 02    CALL    0285             Event Block aushängen
1C8D  D1          POP     DE
1C8E  C1          POP     BC
1C8F  C9          RET

**********************************    KM BREAK EVENT
1C90  21 0C B5    LD      HL,B50C          Flag f. Event Block gültig
1C93  7E          LD      A,(HL)           laden
1C94  36 00       LD      (HL),00          und löschen
1C96  BE          CP      (HL)             Flag war gelöscht?
1C97  C8          RET     Z                dann zurück
1C98  C5          PUSH    BC
1C99  D5          PUSH    DE
1C9A  23          INC     HL               Zeiger auf Break Event Block
1C9B  CD E2 01    CALL    01E2             Break Event einhängen
1C9E  0E EF       LD      C,EF             BRK-Zeichen
1CA0  CD FE 1C    CALL    1CFE             in Ringbuffer schreiben
1CA3  D1          POP     DE
1CA4  C1          POP     BC
1CA5  C9          RET

**********************************    KM GET REPEAT
                                      IN : A: Tastennummer
                                      OUT: Z:=0, wenn Repeat
1CA6  2A 47 B5    LD      HL,(B547)        Adr. d. Repeat Tabelle
1CA9  18 1D       JR      1CC8             Z:=0, wenn Repeat f. Taste

**********************************    KM SET REPEAT
                                      IN : A: Tastennummer
                                            B: Repeat-Flag ($FF f. on)
                                      OUT: CY:=1, wenn o.k.
1CAB  FE 50       CP      50               Tastennummer >79?
1CAD  D0          RET     NC               dann raus, Fehler
1CAE  2A 47 B5    LD      HL,(B547)        Start d. Repeat Tabelle
1CB1  CD CD 1C    CALL    1CCD             entspr. Adr. u Bitmaske holen
1CB4  4F          LD      C,A              Bitmaske retten
1CB5  2F          CPL                        Bit in Tabelle
```

```
1CB6  A6           AND    (HL)          auf Null
1CB7  77           LD     (HL),A        setzen
1CB8  79           LD     A,C          Bitmaske
1CB9  A0           AND    B            entspr. Bit in Flag isolieren
1CBA  B6           OR     (HL)         und restliche Bits aus Tabelle
1CBB  77           LD     (HL),A       neues Tabellenbyte abspeichern
1CBC  C9           RET

**********************************   KM TEST KEY
                                     IN : A: Tastennummer
                                     OUT: C: CTRL/SHIFT-Flag
                                          Z:=0, wenn Taste gedrückt
                                          A:=0, wenn Taste nicht gedr.
1CBD  F5           PUSH   AF           Tastennummer retten
1CBE  3A ED B4     LD     A,(B4ED)     CTRL-SHIFT-Flags
1CC1  E6 A0        AND    A0           isolieren
1CC3  4F           LD     C,A          und nach C retten
1CC4  F1           POP    AF           Tastennummer
1CC5  21 EB B4     LD     HL,B4EB      Start der Key State Map
1CC8  CD CD 1C     CALL   1CCD         entspr. Adr. u. Bitmaske holen
1CCB  A6           AND    (HL)         Bitmaske und Tab.-Byte verkn.
1CCC  C9           RET

**********************************   Adr. und Bitmaske holen
                                     IN : A: Tastennummer
                                          HL: Adr. d. Tabellenanfangs
                                     OUT: A: Bitmaske
                                          HL: Adr. entspr. Tab.-Byte
1CCD  D5           PUSH   DE
1CCE  F5           PUSH   AF           Tastennummer retten
1CCF  E6 F8        AND    F8           unteren 3 Bits eliminieren
1CD1  0F           RRCA                  durch 8 teilen, ergibt
1CD2  0F           RRCA                  die Zeilennummer
1CD3  0F           RRCA                  der Taste
1CD4  5F           LD     E,A          Zeilennummer retten
1CD5  16 00        LD     D,00         Hi-Byte löschen
1CD7  19           ADD    HL,DE        ergibt Adr. d. Tabellenbytes
1CD8  F1           POP    AF           Tastennummer
1CD9  E5           PUSH   HL           Adresse retten
1CDA  21 E5 1C     LD     HL,1CE5      Zeiger auf Bitmasken
1CDD  E6 07        AND    07           untere 3 Bits ergeben Bitno.
1CDF  5F           LD     E,A          Bitnummer
1CE0  19           ADD    HL,DE        zu Start d. Bitmasken addieren
1CE1  7E           LD     A,(HL)       entspr. Bitmaske laden
1CE2  E1           POP    HL           Tabellenadresse vom Stack
1CE3  D1           POP    DE
1CE4  C9           RET

1CE5  01 02 04 08 10 20 40 80          Bitmasken

**********************************   Ringbuffer initialisieren
1CED  F3           DI
1CEE  21 3C B5     LD     HL,B53C      Adresse der Buffer Parameter
1CF1  36 15        LD     (HL),15      Anz. freier Einträge setzen
1CF3  23           INC    HL           Zeiger auf Schreibzeiger
1CF4  AF           XOR    A
1CF5  77           LD     (HL),A       Schreibzeiger setzen
1CF6  23           INC    HL           Zeiger auf Anz. der Einträge+1
```

```
1CF7  36 01      LD    (HL),01      Anz. d. Einträge+1 setzen
1CF9  23         INC   HL           Zeiger auf Lesezeiger
1CFA  77         LD    (HL),A       Lesezeiger setzen
1CFB  23         INC   HL           Zeiger auf Anz. d. Einträge
1CFC  77         LD    (HL),A       Anz. d. Einträge setzen
1CFD  C9         RET

***********************************  Eintrag in Buffer schreiben
                                    IN : BC: Tastenkoordinaten
                                    OUT: CY:=1, wenn o.k.
1CFE  21 3C B5   LD    HL,B53C        Anz. freie Einträge +1
1D01  B7         OR    A              CY:=0 f. evtl. Fehler
1D02  35         DEC   (HL)           Anz. freier Words
1D03  28 0E      JR    Z,1D13         =0? dann Buffer voll, Fehler
1D05  CD 2C 1D   CALL  1D2C           sonst Zeiger in Buffer nach HL
1D08  71         LD    (HL),C           Tastenkoordinaten
1D09  23         INC   HL               in den Ringbuffer
1D0A  70         LD    (HL),B           schreiben
1D0B  21 40 B5   LD    HL,B540        Anz. d. Einträge
1D0E  34         INC   (HL)           erhöhen
1D0F  21 3E B5   LD    HL,B53E        Anz. der Einträge +1
1D12  37         SCF                  CY:=1 f. o.k.
1D13  34         INC   (HL)
1D14  C9         RET

***********************************  Eintrag aus Buffer lesen
                                    OUT: CY:=1, wenn o.k.
                                         BC: Eintrag aus Buffer
1D15  21 3E B5   LD    HL,B53E        Anz. d. Einträge im Buffer+1
1D18  B7         OR    A              CY:=0 f. evtl. Fehler
1D19  35         DEC   (HL)           Anz. d. Einträge im Buffer
1D1A  28 0E      JR    Z,1D2A         Buffer leer? dann Fehler
1D1C  CD 2C 1D   CALL  1D2C           Adresse d. Eintrages berechn.
1D1F  4E         LD    C,(HL)           Eintrag aus
1D20  23         INC   HL               Ringbuffer
1D21  46         LD    B,(HL)           lesen, nach BC
1D22  21 40 B5   LD    HL,B540        Anz. d. Einträge im Buffer
1D25  35         DEC   (HL)           erniedrigen
1D26  21 3C B5   LD    HL,B53C        Anz. d. freien Einträge
1D29  37         SCF                  CY:=1 f. o.k.
1D2A  34         INC   (HL)
1D2B  C9         RET

***********************************  Zeiger in Ringbuffer berechnen
                                    IN : HL: Zeiger vor
                                             Lese-/Schreibzeiger
                                    OUT: HL: Zeiger in Ringbuffer
1D2C  23         INC   HL             Zeiger auf Bufferzeiger
1D2D  34         INC   (HL)           Bufferzeiger erhöhen
1D2E  7E         LD    A,(HL)         und laden
1D2F  FE 14      CP    14             am Bufferende?
1D31  20 02      JR    NZ,1D35        sonst weiter
1D33  AF         XOR   A              ggf. Zeiger auf Bufferanfang
1D34  77         LD    (HL),A         setzen
1D35  87         ADD   A              *2 f. 2 Bytes pro Eintrag
1D36  CE 14      ADC   14               zum Beginn des
1D38  6F         LD    L,A              Ringbuffers (bei $B514)
1D39  CE B5      ADC   B5               addieren
```

```
1D3B  95          SUB    L
1D3C  67          LD     H,A
1D3D  C9          RET
***********************************    KM GET TRANSLATE
                                       IN : A: Tastennummer
                                       OUT: A: ASCII-Code, normal
1D3E  2A 41 B5    LD     HL,(B541)        Start Translation Table
1D41  18 08       JR     1D4B             Code aus Tabelle holen

***********************************    KM GET SHIFT
                                       IN : A: Tastennummer
                                       OUT: A: ASCII-Code, mit Shift
1D43  2A 43 B5    LD     HL,(B543)        Start Shift-Tabelle
1D46  18 03       JR     1D4B             Code aus Tabelle holen

***********************************    KM GET CTRL
                                       IN : A: Tastennummer
                                       OUT: A: ASCII-Code, mit Ctrl
1D48  2A 45 B5    LD     HL,(B545)        Start Control-Tabelle

***********************************    Tastencode holen
                                       IN : A: Tastennummer
                                           HL: Tabellenanfang
                                       OUT: A: Tastencode
1D4B  85          ADD    L                 Tastennummer
1D4C  6F          LD     L,A               zum
1D4D  8C          ADC    H                 Tabellenstart
1D4E  95          SUB    L                 addieren,
1D4F  67          LD     H,A               ergibt Adr. des Codes
1D50  7E          LD     A,(HL)           Code laden
1D51  C9          RET

***********************************    KM SET TRANSLATE
                                       IN : A: Tastennummer
                                            B: Tastencode, normal
1D52  2A 41 B5    LD     HL,(B541)        Start Translation Table
1D55  18 08       JR     1D5F             Code in Tabelle schreiben

***********************************    KM SET SHIFT
                                       IN : A: Tastennummer
                                            B: Tastencode, mit Shift
1D57  2A 43 B5    LD     HL,(B543)        Start Shift-Tabelle
1D5A  18 03       JR     1D5F             Code in Tabelle schreiben

***********************************    KM SET CTRL
                                       IN : A: Tastennummer
                                            B: Tastencode, mit Ctrl
1D5C  2A 45 B5    LD     HL,(B545)        Start Control-Tabelle

***********************************    Tastencode setzen
                                       IN : A: Tastennummer
                                            B: entspr. Tastencode
                                            HL: Start der Tabelle
1D5F  FE 50       CP     50               Tastennummer>79?
1D61  D0          RET    NC               dann Fehler, CY:=0, zurück
1D62  85          ADD    L                 Tastennummer
1D63  6F          LD     L,A               zum Start der
1D64  8C          ADC    H                 Tabelle
```

```
1D65  95          SUB   L              addieren,
1D66  67          LD    H,A            ergibt Adr. des Codes
1D67  70          LD    (HL),B       Code setzen
1D68  C9          RET

*********************************** KEY TRANSLATION TABLE
1D69  F0 F3 F1 89 86 83 8B 8A
1D71  F2 E0 87 88 85 81 82 80
1D79  10 5B 0D 5D 84 FF 5C FF              \
1D81  5E 2D 40 70 3B 3A 2F 2E       ^-@p;:/.
1D89  30 39 6F 69 6C 6B 6D 2C       09oilkm,
1D91  38 37 75 79 68 6A 6E 20       87uyhjn
1D99  36 35 72 74 67 66 62 76       65rtgfbv
1DA1  34 33 65 77 73 64 63 78       43ewsdcx
1DA9  31 32 FC 71 09 61 FD 7A       12 q a z
1DB1  0B 0A 08 09 58 5A FF 7F           XZ

*********************************** KEY SHIFT TABLE
1DB9  F4 F7 F5 89 86 83 8B 8A
1DC1  F6 E0 87 88 85 81 82 80
1DC9  10 7B 0D 7D 84 FF 60 FF
1DD1  A3 3D 7C 50 2B 2A 3F 3E        = P+*?>
1DD9  5F 29 4F 49 4C 4B 4D 3C        )OILKM<
1DE1  28 27 55 59 48 4A 4E 20       ('UYHJN
1DE9  26 25 52 54 47 46 42 56       &%RTGFBV
1DF1  24 23 45 57 53 44 43 58       $#EWSDCX
1DF9  21 22 FC 51 09 41 FD 5A       !" Q A Z
1E01  0B 0A 08 09 58 5A FF 7F           XZ

*********************************** KEY CTRL TABLE
1E09  F8 FB F9 89 86 83 8C 8A
1E11  FA E0 87 88 85 81 82 80
1E19  10 1B 0D 1D 84 FF 1C FF
1E21  1E FF 00 10 FF FF FF FF
1E29  1F FF 0F 09 0C 0B 0D FF
1E31  FF FF 15 19 08 0A 0E FF
1E39  FF FF 12 14 07 06 02 16
1E41  FF FF 05 17 13 04 03 18
1E49  FF 7E FC 11 E1 01 FE 1A
1E51  FF FF FF FF FF FF FF 7F

1E59  07 03 4B FF FF FF FF FF
1E61  AB 8F

1E63  C7          RST   00
1E64  C7          RST   00
1E65  C7          RST   00
1E66  C7          RST   00
1E67  C7          RST   00
```

```
----------------------- SOUND MANAGER (SOUND) ----------------------

*********************************   SOUND RESET
1E68  AF          XOR    A
1E69  F3          DI
1E6A  32 52 B5    LD     (B552),A     laufende und
1E6D  32 51 B5    LD     (B551),A     alte  Aktivitäten löschen
1E70  21 55 B5    LD     HL,B555      Adresse des Sound Event Blocks
1E73  11 03 1F    LD     DE,1F03      Zeiger auf Event Routine
1E76  06 81       LD     B,81         Event asynchr., Near Address
1E78  CD D2 01    CALL   01D2         Event Block aufbauen
1E7B  3E 3F       LD     A,3F         Kontrollbyte für PSG
1E7D  32 19 B6    LD     (B619),A     setzen
1E80  21 5C B5    LD     HL,B55C      Paramaterblöcke der Kanäle A-C
1E83  01 3D 00    LD     BC,003D      Länge der Params -2
1E86  11 08 01    LD     DE,0108      Werte für Kanal- u. Rauschmaske
1E89  AF          XOR    A            Null
1E8A  77          LD     (HL),A       Kanalnummer setzen
1E8B  23          INC    HL           Zeiger auf Kanalmaske
1E8C  72          LD     (HL),D       Kanalmaske setzen
1E8D  23          INC    HL           Zeiger auf Rauschmaske
1E8E  73          LD     (HL),E       Rauschmaske setzen
1E8F  09          ADD    HL,BC        restlichen Block übergehen
1E90  3C          INC    A            Kanalnummer erhöhen
1E91  EB          EX     DE,HL          Kanalmaske und
1E92  29          ADD    HL,HL          Rauschmaske
1E93  EB          EX     DE,HL          nach links verschieben
1E94  FE 03       CP     03           Kanalnummer <2?
1E96  20 F2       JR     NZ,1E8A      dann nächsten Kanal initialisieren
1E98  0E 07       LD     C,07         b0-b2:=1, f. alle Blöcke

*********************************   Parameter-Blöcke initialisieren
                                     IN : C: Kanalbits
1E9A  DD E5       PUSH   IX
1E9C  E5          PUSH   HL
1E9D  21 1D B5    LD     HL,B51D      Start Params A -$3F
1EA0  41          LD     B,C          Kanalbits
1EA1  11 3F 00    LD     DE,003F      Länge eines Blockes
1EA4  19          ADD    HL,DE        addieren
1EA5  CB 38       SRL    B            bis ein 1-Bit gefunden
1EA7  30 F8       JR     NC,1EA1
1EA9  C5          PUSH   BC           restliche Kanalbits retten
1EAA  E5          PUSH   HL           Zeiger auf Params
1EAB  DD E1       POP    IX           nach IX
1EAD  EB          EX     DE,HL        und nach DE
1EAE  CD 7F 22    CALL   227F         Kanal ausschalten
1EB1  13          INC    DE             Zeiger auf Kanalstatus
1EB2  13          INC    DE             berechnen
1EB3  13          INC    DE
1EB4  6B          LD     L,E            und nach
1EB5  62          LD     H,D            HL kopieren
1EB6  13          INC    DE           Zeiger auf ENT-Flag
1EB7  01 3B 00    LD     BC,003B      Anzahl der Bytes bis Blockende
1EBA  36 00       LD     (HL),00      mit Nullen
1EBC  ED B0       LDIR                füllen
1EBE  DD 36 1C 04 LD     (IX+1C),04   Anzahl der freien Datenblöcke
1EC2  C1          POP    BC           Kanal- u. Rauschmaske
1EC3  EB          EX     DE,HL        Zeiger aus nächsten Block -> HL
```

```
1EC4  04           INC    B             zum Ausgleich f. Herunterzähl.
1EC5  10 DE        DJNZ   1EA5          ggf. restliche Blöcke bearbeiten
1EC7  E1           POP    HL
1EC8  DD E1        POP    IX
1ECA  C9           RET

********************************** SOUND HOLD
                                  OUT: CY:=1, wenn o.k.
                                          CY:=0, w. keine lfd. Aktiv.
1ECB  21 52 B5     LD     HL,B552       Zeiger auf lfd. Aktivitäten
1ECE  F3           DI
1ECF  7E           LD     A,(HL)        lfd. Aktivitäten laden
1ED0  36 00        LD     (HL),00       und löschen
1ED2  FB           EI
1ED3  B7           OR     A             keine lfd. Aktivitäten?
1ED4  C8           RET    Z             dann CY:=0, raus
1ED5  2B           DEC    HL              sonst lfd. Aktivitäten
1ED6  77           LD     (HL),A          als alte Aktivitäten
1ED7  2E 03        LD     L,03          Zähler f. 3 Kanäle
1ED9  0E 00        LD     C,00          Lautstärke:=0, d.h. ausschalt.
1EDB  3E 07        LD     A,07            7+Kanalnummer
1EDD  85           ADD    L               = entspr. Lautstärkeregister
1EDE  CD 26 08     CALL   0826          Lautstärke setzen
1EE1  2D           DEC    L             Kanalnummer
1EE2  20 F7        JR     NZ,1EDB       noch nicht =0? nächsten Kanal
1EE4  37           SCF                  CY:=1 f. o.k.
1EE5  C9           RET

********************************** SOUND CONTINUE
1EE6  3A 51 B5     LD     A,(B551)      alte Aktivitäten
1EE9  B7           OR     A             keine?
1EEA  C8           RET    Z             dann raus
1EEB  DD 21 1D B5  LD     IX,B51D       Adr. Params A -$3F
1EEF  11 3F 00     LD     DE,003F       Länge eines Blockes
1EF2  DD 19        ADD    IX,DE         addieren
1EF4  CB 3F        SRL    A             bis ein 1-Bit gefunden
1EF6  F5           PUSH   AF            restl. Kanalmaske retten
1EF7  DD 7E 0F     LD     A,(IX+0F)     Lautstärke aus Params
1EFA  DC 76 22     CALL   C,2276        ggf. setzen
1EFD  F1           POP    AF            restl. Kanalmaske
1EFE  20 F2        JR     NZ,1EF2       noch Kanäle? d. bearbeiten
1F00  C3 1E 20     JP     201E          alte u. lfd. Aktiv. setzen

********************************** Sound Event (asynchronous)
1F03  DD E5        PUSH   IX
1F05  21 50 B5     LD     HL,B550       Zeiger auf restl. Aktiv.
1F08  E5           PUSH   HL            retten
1F09  AF           XOR    A             restl. Aktiv.
1F0A  77           LD     (HL),A        löschen
1F0B  23           INC    HL            alte Aktivitäten
1F0C  46           LD     B,(HL)        laden
1F0D  C5           PUSH   BC            und retten
1F0E  23           INC    HL            laufende Aktivitäten
1F0F  B6           OR     (HL)          laden u. Flags setzen
1F10  28 22        JR     Z,1F34        keine lfd. Aktiv.? d. alte
1F12  DD 21 1D B5  LD     IX,B51D       Start Params A -$3F
1F16  01 3F 00     LD     BC,003F       Länge eines Blockes
1F19  DD 09        ADD    IX,BC         addieren
```

```
1F1B  CB 3F        SRL   A            bis ein aktiver
1F1D  30 FA        JR    NC,1F19      Block gefunden
1F1F  F5           PUSH  AF           restl. Kanalmaske retten
1F20  DD 7E 04     LD    A,(IX+04)    Flag f. ENT-Folge schwebend
1F23  1F           RRA                ins Carry
1F24  DC C2 22     CALL  C,22C2       ggf. ENT-Folge bearbeiten
1F27  DD 7E 07     LD    A,(IX+07)    Flag f. ENV-Folge schwebend
1F2A  1F           RRA                ins Carry
1F2B  DC B6 21     CALL  C,21B6       ggf. ENV-Folge bearbeiten
1F2E  DC A8 20     CALL  C,20A8       Tonende? d. nächster Eintrag
1F31  F1           POP   AF           restl. Kanalmaske
1F32  20 E2        JR    NZ,1F16      noch Kanäle? d. bearbeiten
1F34  C1           POP   BC           alte Aktivitäten
1F35  E1           POP   HL           restliche Aktivitäten
1F36  7E           LD    A,(HL)       laden
1F37  B7           OR    A            keine Aktivitäten mehr übrig?
1F38  28 20        JR    Z,1F5A       dann raus
1F3A  4F           LD    C,A          restl. Aktiv. retten
1F3B  23           INC   HL           alte Aktiv.
1F3C  7E           LD    A,(HL)       laden
1F3D  70           LD    (HL),B       urspr. alte Aktiv. setzen
1F3E  A8           XOR   B            Änderungen d. alten Aktiv.
1F3F  47           LD    B,A          nach B
1F40  23           INC   HL           lfd. Aktivitäten
1F41  B6           OR    (HL)         u. die Änderungen d. alten
1F42  77           LD    (HL),A       zusammen als lfd. setzen
1F43  78           LD    A,B          Änderungen der alten Aktiv.
1F44  2F           CPL                aus restl. Aktiv.
1F45  A1           AND   C            ausmaskieren
1F46  28 12        JR    Z,1F5A       keine Aktiv. übrig? dann raus
1F48  DD 21 1D B5  LD    IX,B51D        ansonsten alle restl.
1F4C  11 3F 00     LD    DE,003F        Aktiv. ohne Änderung d.
1F4F  DD 19        ADD   IX,DE          alten Aktivitäten
1F51  CB 3F        SRL   A              deaktivieren
1F53  F5           PUSH  AF             und abschalten
1F54  DC 7F 22     CALL  C,227F
1F57  F1           POP   AF
1F58  20 F5        JR    NZ,1F4F
1F5A  AF           XOR   A            Null als Zeichen f.
1F5B  32 54 B5     LD    (B554),A     Aktivitäten bearbeitet setzen
1F5E  DD E1        POP   IX
1F60  C9           RET

*********************************** Scan Sound Queues
1F61  21 52 B5     LD    HL,B552        lfd. Sound Aktivitäten
1F64  7E           LD    A,(HL)       laden
1F65  B7           OR    A            Sound inaktiv?
1F66  C8           RET   Z            dann zurück
1F67  23           INC   HL           Zeiger a. 100Hz Frequenzteiler
1F68  35           DEC   (HL)         Interrupt-Zähler dekrement.
1F69  C0           RET   NZ           zurück, wenn noch nicht null
1F6A  34           INC   (HL)         sonst auf 1 setzen
1F6B  23           INC   HL           Flag f. 'Folgen bearbeitet'
1F6C  7E           LD    A,(HL)       laden
1F6D  B7           OR    A            Folgen noch unbearbeitet?
1F6E  C0           RET   NZ           dann zurück
1F6F  2B           DEC   HL           Freqenzteiler
1F70  36 03        LD    (HL),03      auf 3 setzen (300Hz/3=100Hz)
```

```
1F72  2B           DEC   HL          lfd. Aktivitäten
1F73  46           LD    B,(HL)      laden
1F74  21 22 B5     LD    HL,B522     Zeiger ENT-Pausenzeit A -$3F
1F77  11 3F 00     LD    DE,003F     Länge eines Param-Blockes
1F7A  AF           XOR   A           Flag f. aktive ENV/ENT-Folgen
1F7B  19           ADD   HL,DE       Blocklänge addieren
1F7C  CB 38        SRL   B           bis aktiver Kanal
1F7E  30 FB        JR    NC,1F7B     gefunden
1F80  35           DEC   (HL)        ENV-Pausenzeit
1F81  20 05        JR    NZ,1F88     noch nicht zuende? d. weiter
1F83  2B           DEC   HL          sonst Flag f. aktive ENV-Folge
1F84  CB 06        RLC   (HL)        ins Carry und b0 d. Flags
1F86  8A           ADC   D           Flag addieren (D=$00)
1F87  23           INC   HL          Zeiger auf ENV-Pausenzähler
1F88  23           INC   HL          Zeiger auf ENT-Pausenzähler
1F89  35           DEC   (HL)        diesen herunterzählen
1F8A  20 05        JR    NZ,1F91     Pause noch nicht zuende?
1F8C  23           INC   HL          sonst Flag f. aktive ENT-Folge
1F8D  CB 06        RLC   (HL)        ins Carry und in b0
1F8F  8A           ADC   D           und zum Akku addieren (D=$00)
1F90  2B           DEC   HL          Zeiger auf ENT-Pausenzähler
1F91  2B           DEC   HL          Zeiger auf ENV-Pausenzähler
1F92  04           INC   B           Kanalmaske z. Ausgl. erhöhen
1F93  10 E6        DJNZ  1F7B        noch Kanäle? d. bearbeiten
1F95  B7           OR    A           irgendwelche Folgen zu bearb.?
1F96  C8           RET   Z           sonst zurück
1F97  21 54 B5     LD    HL,B554     Flag f. Folgen bearbeiten
1F9A  77           LD    (HL),A      setzen
1F9B  23           INC   HL          Zeiger auf Sound Event
1F9C  C3 E2 01     JP    01E2        Sound Event einhängen

***********************************  SOUND QUEUE
                                     IN : (HL)..(HL+8): Übergabe-Block
                                              0: Kanalstatus
                                              1: No. d. ENV-Folge
                                              2: No. d. ENT-Folge
                                              3/4: Tonperiode
                                              5: Rauschperiode
                                              6: Lautstärke
                                              7/8: Tondauer
                                     OUT: CY:=1, wenn o.k.
                                          CY:=0, wenn Queue voll
1F9F  CD E6 1E     CALL  1EE6        alte Aktiv. wieder einsetzen
1FA2  7E           LD    A,(HL)      Kanalstatus aus Übergabe
1FA3  E6 07        AND   07          Kanalbits isolieren
1FA5  37           SCF               CY:=1 f. evtl. o.k.
1FA6  C8           RET   Z           kein Kanal? dann o.k., raus
1FA7  4F           LD    C,A         Kanalbits retten
1FA8  B6           OR    (HL)        restlichen Status laden
1FA9  FC 9A 1E     CALL  M,1E9A      flush? d. Param-Block löschen
1FAC  41           LD    B,C         Kanalbits
1FAD  DD 21 1D B5  LD    IX,B51D     Start Params A -$3F
1FB1  11 3F 00     LD    DE,003F     Länge eines Blockes
1FB4  AF           XOR   A
1FB5  DD 19        ADD   IX,DE       addieren
1FB7  CB 38        SRL   B           bis entspr. Kanal
1FB9  30 FA        JR    NC,1FB5     gefunden
1FBB  DD 72 1E     LD    (IX+1E),D   Event-Adr. löschen
```

```
1FBE  DD BE 1C     CP     (IX+1C)     noch Einträge frei?
1FC1  3F           CCF                dann CY:=0
1FC2  9F           SBC    A           bzw. A:=$00
1FC3  04           INC    B           zum Ausgleich erhöhen
1FC4  10 EF        DJNZ   1FB5        noch Kanäle? dann bearbeiten
1FC6  B7           OR     A           noch Einträge frei?
1FC7  C0           RET    NZ          sonst zurück, CY:=0 f. Fehler
1FC8  41           LD     B,C         Kanalbits
1FC9  7E           LD     A,(HL)      Status aus Übergabe
1FCA  1F           RRA                  Zielkanal
1FCB  1F           RRA                  aus Status
1FCC  1F           RRA                  heraus
1FCD  B0           OR     B           und noch als Rendezvous setzen
1FCE  E6 0F        AND    0F          u. isolieren, 'hold' in b3
1FD0  4F           LD     C,A         neuer Status nach C
1FD1  23           INC    HL          Zeiger auf ENV-Folgenummer
1FD2  DD 21 1D B5  LD     IX,B51D     Start Params A -$3F
1FD6  11 3F 00     LD     DE,003F     Länge eines Blockes
1FD9  DD 19        ADD    IX,DE       addieren
1FDB  CB 38        SRL    B           bis ein Zielkanal
1FDD  30 FA        JR     NC,1FD9     gefunden
1FDF  E5           PUSH   HL          Zeiger in Übergabe retten
1FE0  C5           PUSH   BC          Kanalbits u. Status retten
1FE1  DD 7E 1B     LD     A,(IX+1B)   No. d. nächst. freien Eintrags
1FE4  DD 34 1B     INC    (IX+1B)     erhöhen
1FE7  DD 35 1C     DEC    (IX+1C)     Anz. freier Einträge dekrem.
1FEA  EB           EX     DE,HL       Zeiger in Übergabe nach DE
1FEB  CD 3A 20     CALL   203A        Adr. d. Daten-Blockes berechn.
1FEE  E5           PUSH   HL          und retten
1FEF  EB           EX     DE,HL       Zeiger in Übergabe
1FF0  DD 7E 01     LD     A,(IX+01)   Kanalmaske
1FF3  2F           CPL                  aus Status löschen
1FF4  A1           AND    C             (kein Rendezvous mit sich)
1FF5  12           LD     (DE),A      und als Datenstatus setzen
1FF6  13           INC    DE          Zeiger f. Daten weiter
1FF7  7E           LD     A,(HL)      ENV-Folgenummer
1FF8  23           INC    HL          Zeiger auf ENT-Folgenummer
1FF9  87           ADD    A             Nummer d. ENV-Folge
1FFA  87           ADD    A             in obere vier Bits
1FFB  87           ADD    A             schieben
1FFC  87           ADD    A
1FFD  47           LD     B,A         und nach B retten
1FFE  7E           LD     A,(HL)      ENT-Folgenummer laden
1FFF  23           INC    HL          Zeiger auf Tonperiode
2000  E6 0F        AND    0F          untere 4 Bits isolieren
2002  B0           OR     B           ENV-Folgeno. in oberes Nibble
2003  12           LD     (DE),A      in Datenblock speichern
2004  13           INC    DE          Zeiger weiter
2005  01 06 00     LD     BC,0006     Anz. d. restl. Datenbytes
2008  ED B0        LDIR               restl. Übergabe in Datenblock
200A  E1           POP    HL          Adr. d. Datenblocks
200B  F3           DI
200C  DD 7E 1A     LD     A,(IX+1A)   Anz. d. Einträge in Queue
200F  DD 34 1A     INC    (IX+1A)     erhöhen
2012  DD B6 03     OR     (IX+03)     nur 1 Block, Queue inaktiv?
2015  FB           EI
2016  CC BD 20     CALL   Z,20BD      dann Kanal aktivieren
2019  C1           POP    BC          Kanalbits u. Datenstatus
```

```
201A  E1           POP   HL         Zeiger auf Übergabe (ENV-No.)
201B  04           INC   B          zum Ausgleich erhöhen
201C  10 B8        DJNZ  1FD6       noch Kanäle? d. bearbeiten
201E  E5           PUSH  HL         Zeiger auf Übergabe
201F  21 51 B5     LD    HL,B551    alte Aktivitäten
2022  7E           LD    A,(HL)     laden
2023  B7           OR    A          keine alten Aktivitäten?
2024  28 11        JR    Z,2037     dann CY:=1 f. o.k., raus
2026  36 00        LD    (HL),00    alte Aktivitäten löschen
2028  F3           DI
2029  23           INC   HL         laufende Aktivitäten
202A  46           LD    B,(HL)     laden, nach B
202B  B0           OR    B          zusammen mit alten Aktiv.
202C  77           LD    (HL),A     als lfd. neu setzen
202D  78           LD    A,B        lfd. Aktivitäten
202E  B7           OR    A          keine?
202F  20 05        JR    NZ,2036    sonst raus
2031  23           INC   HL         100 Hz Frequenzteiler
2032  36 03        LD    (HL),03    auf Startwert setzen
2034  23           INC   HL         Flag f. Folgen schwebend
2035  77           LD    (HL),A     löschen: Folgen bearbeitet
2036  FB           EI
2037  E1           POP   HL         Zeiger auf Übergabe
2038  37           SCF              CY:=1 f. o.k.
2039  C9           RET

***********************************  Datenblock Adresse berechnen
                                     IN : IX: Zeiger auf lfd. Params
                                          A<1..0>: Datenblock-Nummer
                                     OUT: HL: Adresse d. Datenblocks
203A  E6 03        AND   03             untere Bits isolieren
203C  87           ADD   A                Datenblocknummer
203D  87           ADD   A                mit 8 (f. 8 Bytes/Block)
203E  87           ADD   A                multiplizieren
203F  C6 1F        ADD   1F             Offset in Params addieren
2041  DD E5        PUSH  IX             Zeiger auf Params
2043  E1           POP   HL             nach HL
2044  85           ADD   L                Offset auf Parameterstart
2045  6F           LD    L,A              zum Parameterstart
2046  8C           ADC   H                addieren,
2047  95           SUB   L                ergibt absolute Adresse
2048  67           LD    H,A              in HL
2049  C9           RET

***********************************  SOUND RELEASE
                                     IN : A: Kanalmaske
204A  6F           LD    L,A            Kanalbits retten
204B  CD E6 1E     CALL  1EE6           alte Aktivitäten wieder an
204E  7D           LD    A,L            Kanalbits zurück
204F  E6 07        AND   07             und isolieren
2051  C8           RET   Z              keine Kanäle? dann raus
2052  DD 21 1D B5  LD    IX,B51D        Start Params A -$3F
2056  11 3F 00     LD    DE,003F        Länge eines Blockes
2059  DD 19        ADD   IX,DE          addieren
205B  CB 3F        SRL   A              bis entspr. Kanalbit
205D  30 FA        JR    NC,2059        gefunden
205F  F5           PUSH  AF             restl. Kanalbits retten
2060  DD CB 03 5E  BIT   3,(IX+03)      Queue im Haltezustand?
```

```
2064  C4 B7 20     CALL   NZ,20B7       d. Kanal aktivieren
2067  F1           POP    AF            restl. Kanalbits
2068  20 EC        JR     NZ,2056       noch Kanäle? d. bearbeiten
206A  18 B2        JR     201E          Aktivitäten neu setzen, raus

***********************************  SOUND CHECK
                                     IN : A: Kanalmaske
                                     OUT: A: Kanalstatus
206C  E6 07        AND    07            Kanalbits isolieren
206E  C8           RET    Z             keine Kanäle? d. raus
206F  21 20 B5     LD     HL,B520       Zeiger Status A -$3F
2072  11 3F 00     LD     DE,003F       Länge eines Blockes
2075  19           ADD    HL,DE         addieren
2076  1F           RRA                  bis 1. Kanalbit
2077  30 FC        JR     NC,2075       gefunden
2079  F3           DI
207A  7E           LD     A,(HL)        Status d. Kanals laden
207B  87           ADD    A               und nach oben schieben,
207C  87           ADD    A               um Anz. freier Queue-
207D  87           ADD    A               Einträge noch einzutragen
207E  11 19 00     LD     DE,0019       Anz. d. Bytes b. freie Eintr.
2081  19           ADD    HL,DE         addieren
2082  B6           OR     (HL)          Anz. freier Plätze in Queue
2083  23           INC    HL              Zeiger auf
2084  23           INC    HL              Event-Adr., Hi-Byte
2085  36 00        LD     (HL),00       Event löschen aus Params
2087  FB           EI
2088  C9           RET

***********************************  SOUND ARM EVENT
                                     IN : A: Kanalbits
                                          HL: Adr. d. Event-Blocks
2089  E6 07        AND    07            Kanalbits isolieren
208B  C8           RET    Z             kein Kanal? dann raus
208C  EB           EX     DE,HL         Zeiger auf Event retten
208D  21 39 B5     LD     HL,B539       Anz. Datenblöcke Kanal A -$3F
2090  01 3F 00     LD     BC,003F       Länge eines Parameterblocks
2093  09           ADD    HL,BC         addieren
2094  1F           RRA                  bis ein Kanal
2095  30 FC        JR     NC,2093       gefunden
2097  AF           XOR    A
2098  F3           DI
2099  BE           CP     (HL)          noch freie Plätze in Queue?
209A  23           INC    HL            Zeiger auf Event-Adresse
209B  73           LD     (HL),E        Lo-Byte d. Adr. speichern
209C  23           INC    HL            Zeiger auf Hi-Byte der Adr.
209D  20 03        JR     NZ,20A2       noch Plätze frei? d. einhängen
209F  72           LD     (HL),D        sonst auch Hi-Byte setzen
20A0  FB           EI
20A1  C9           RET

20A2  77           LD     (HL),A        Hi-Byte löschen
20A3  FB           EI
20A4  EB           EX     DE,HL         Adr. d. Event-Blocks nach HL
20A5  C3 E2 01     JP     01E2          Event einhängen
```

```
*********************************** nächsten Queue-Eintrag
                                    IN : IX: Zeiger lfd. Params
20A8  DD 7E 1A     LD    A,(IX+1A)     Anz. d. Einträge in Queue
20AB  B7           OR    A             keine Einträge mehr?
20AC  CA 7F 22     JP    Z,227F        dann raus, Kanal ausschalten
20AF  DD 7E 01     LD    A,(IX+01)     Kanalmaske
20B2  21 50 B5     LD    HL,B550       in restl. Aktivitäten
20B5  B6           OR    (HL)          hinein
20B6  77           LD    (HL),A        und neu setzen

*********************************** Kanal aktivieren
                                    IN : IX: Zeiger lfd. Params
20B7  DD 7E 19     LD    A,(IX+19)     lfd. Datenblocknummer
20BA  CD 3A 20     CALL  203A          Adr. d. Blocks berechnen
20BD  7E           LD    A,(HL)        Status d. Daten
20BE  B7           OR    A             keine Rendezvous?
20BF  28 0C        JR    Z,20CD        dann Kanal einfach aktivieren
20C1  CB 5F        BIT   3,A           Daten im Haltezustand?
20C3  20 53        JR    NZ,2118       dann Kanal (Queue) anhalten
20C5  E5           PUSH  HL            Zeiger auf Datenblock
20C6  36 00        LD    (HL),00       Datenstatus löschen
20C8  CD 1F 21     CALL  211F          Rendezvous-Bits auswerten
20CB  E1           POP   HL            Zeiger auf Datenblock
20CC  D0           RET   NC            keine Rendezv. b. Unter-Kanal?
20CD  DD 36 03 10  LD    (IX+03),10    sonst Kanal aktivieren
20D1  23           INC   HL            Zeiger auf Folgennummern
20D2  7E           LD    A,(HL)        laden
20D3  E6 F0        AND   F0            No. d. ENV-Folge isolieren
20D5  F5           PUSH  AF            und retten
20D6  AE           XOR   (HL)          No. d. ENT-Folge isolieren
20D7  5F           LD    E,A           und nach E retten
20D8  23           INC   HL            Zeiger auf Tonperiode
20D9  4E           LD    C,(HL)          Tonperiode
20DA  23           INC   HL              nach D,C
20DB  56           LD    D,(HL)          laden
20DC  23           INC   HL            Zeiger auf Rauschperiode
20DD  B2           OR    D               keine ENT-Folge und
20DE  B1           OR    C               Tonperiode =0?
20DF  28 08        JR    Z,20E9        dann weiter
20E1  E5           PUSH  HL            Zeiger auf Rauschperiode
20E2  CD AB 22     CALL  22AB          ENT-Ende d. lfd. Tons bearb.
20E5  DD 56 01     LD    D,(IX+01)     lfd. Kanalmaske
20E8  E1           POP   HL            Zeiger auf Rauschperiode
20E9  4E           LD    C,(HL)        Rauschperiode laden
20EA  23           INC   HL            Lautstärke (Anfangswert)
20EB  5E           LD    E,(HL)        nach E laden
20EC  23           INC   HL            Zeiger auf Tondauer
20ED  7E           LD    A,(HL)          Tondauer
20EE  23           INC   HL              nach
20EF  66           LD    H,(HL)          HL
20F0  6F           LD    L,A             laden
20F1  F1           POP   AF            ENV-Nummer
20F2  CD 75 21     CALL  2175          Dauer, Rauschen u. ENV setz.
20F5  21 51 B5     LD    HL,B551       alte Aktivitäten
20F8  DD 7E 01     LD    A,(IX+01)     lfd. Kanalmaske
20FB  B6           OR    (HL)          lfd. Kanal in alte Aktiv.
20FC  77           LD    (HL),A        setzen
20FD  DD 34 19     INC   (IX+19)       lfd. Datenblocknummer erhöhen
```

```
2100  DD 35 1A     DEC    (IX+1A)       Anz. d. Datenblöcke erniedrig.
2103  DD 34 1C     INC    (IX+1C)       Anz. d freien Blöcke erhöhen
2106  F3           DI
2107  DD 7E 1E     LD     A,(IX+1E)     Hi-Byte d. Event-Adr.
210A  DD 36 1E 00  LD     (IX+1E),00    Event löschen
210E  FB           EI
210F  B7           OR     A             war Event gelöscht?
2110  C8           RET    Z             dann raus
2111  67           LD     H,A             sonst Adr. d. Events
2112  DD 6E 1D     LD     L,(IX+1D)       nach HL laden
2115  C3 E2 01     JP     01E2          und Event einhängen

***********************************   Queue in Haltezustand
                                      IN : IX: Zeiger auf Params
                                             HL: Zeiger auf lfd. Daten
2118  CB 9E        RES    3,(HL)        Hold-Bit in Daten ausschalten
211A  DD 36 03 08  LD     (IX+03),08    Queue in Haltezustand
211E  C9           RET

***********************************   Rendezvous-Bits auswerten
                                      IN : IX: Zeiger auf Params
                                              A: Datenstatus
                                      OUT: CY:=1, wenn Rekursion
211F  DD E5        PUSH   IX              Param-Zeiger retten
2121  47           LD     B,A             Datenstatus
2122  DD 4E 01     LD     C,(IX+01)       Kanalmaske d. Ur-Kanals
2125  DD 21 5C B5  LD     IX,B55C         Param-Zeiger, Kanal A
2129  CB 47        BIT    0,A             wenn Rendezvous mit
212B  20 0C        JR     NZ,2139         Kanal A
212D  DD 21 9B B5  LD     IX,B59B         Param-Zeiger, Kanal B
2131  CB 4F        BIT    1,A             wenn Rendezvous mit
2133  20 04        JR     NZ,2139         Kanal B
2135  DD 21 DA B5  LD     IX,B5DA         sonst Param-Zeiger, Kanal C
2139  F3           DI
213A  DD 7E 03     LD     A,(IX+03)       Status, 1. Unter-Kanal
213D  A1           AND    C               mit Ur-Kanalmaske verknüpfen
213E  28 2D        JR     Z,216D          kein Rend. m. Ur-Kanal?
2140  78           LD     A,B             Ur-Datenstatus
2141  DD BE 01     CP     (IX+01)         Rendezv. nur m. 1. Unter-Kan.?
2144  28 1A        JR     Z,2160          d. Unter-Kanal anschalten
2146  DD E5        PUSH   IX              Zeiger auf 1. Unterkanal
2148  DD 21 DA B5  LD     IX,B5DA         Zeiger auf Kanal C
214C  CB 57        BIT    2,A             wenn auch noch Rendezv. m.
214E  20 04        JR     NZ,2154         Kanal C
2150  DD 21 9B B5  LD     IX,B59B         sonst Zeiger auf Kanal B
2154  DD 7E 03     LD     A,(IX+03)       2. Unterkanal-Status
2157  A1           AND    C               m. Ur-Kanalmaske verknüpfen
2158  28 12        JR     Z,216C          dann nur 2. Unterkanal aktiv.
215A  FB           EI
215B  CD B7 20     CALL   20B7            2. Unterkanal aktivieren
215E  DD E1        POP    IX              Zeiger auf 1. Unterkanal
2160  DD 36 03 00  LD     (IX+03),00      Kanal ausschalten
2164  FB           EI
2165  CD B7 20     CALL   20B7            Kanal aktivieren
2168  DD E1        POP    IX              Zeiger auf Ur-Kanal
216A  37           SCF                    CY:=1 f. Rekursion
216B  C9           RET
216C  E1           POP    HL              Zeiger auf Ur-Daten
```

```
216D  DD E1         POP     IX            Zeiger auf Ur-Kanal
216F  DD 70 03      LD      (IX+03),B     Datenstatus als Kanalstatus
2172  FB            EI
2173  B7            OR      A             CY:=0 f. Rekursionsende
2174  C9            RET

*********************************          Tondauer, Rauschen u. ENV setzen
                                           IN : HL: Tondauer
                                                 E: Lautstärke
                                                 C: Rauschperiode
                                                 D: Kanalmaske
                                                 A<7-4>: ENV-Folgenummer
2175  CB FB         SET     7,E             b7 d. Lautstärke :=1
2177  DD 73 0F      LD      (IX+0F),E     und als Lautstärke setzen
217A  5F            LD      E,A           ENV-Folgenummer retten
217B  7D            LD      A,L             Tondauer
217C  B4            OR      H               nicht null?
217D  20 01         JR      NZ,2180       dann weiter
217F  2B            DEC     HL            sonst Tondauer := $FFFF
2180  DD 75 08      LD      (IX+08),L       Tondauer
2183  DD 74 09      LD      (IX+09),H       setzen
2186  79            LD      A,C           Rauschperiode
2187  B7            OR      A             kein Rauschen?
2188  28 08         JR      Z,2192        dann weiter
218A  3E 06         LD      A,06          sonst als Rauschperiode
218C  CD 26 08      CALL    0826          in den PSG setzen
218F  DD 7E 02      LD      A,(IX+02)     Rauschmaske
2192  B2            OR      D             zusammen m. Kanalmaske
2193  CD 8B 22      CALL    228B          Kanal u. Rauschen anschalten
2196  7B            LD      A,E           ENV-Nummer
2197  B7            OR      A             =0?
2198  28 0A         JR      Z,21A4        dann Default-ENV-Folge setzen
219A  21 0A B6      LD      HL,B60A       Tabellenstart d. ENV-Kurven
219D  16 00         LD      D,00          Hi-Byte löschen
219F  19            ADD     HL,DE         ENV-No.*16 addieren
21A0  7E            LD      A,(HL)        Länge der Folge
21A1  B7            OR      A             Null?
21A2  20 03         JR      NZ,21A7       sonst weiter
21A4  21 B2 21      LD      HL,21B2       Default f. ENV-Folge
21A7  DD 75 0A      LD      (IX+0A),L       Adr. d. lfd.
21AA  DD 74 0B      LD      (IX+0B),H       ENV-Folge setzen
21AD  CD 65 22      CALL    2265          ENV-Werte initialisieren
21B0  18 0D         JR      21BF          1. ENV-Gruppe bearbeiten

*********************************          Default ENV-Folge
21B2  01                                    Länge d. Folge
21B5  01 00  C8

*********************************          lfd. ENV-Gruppe bearbeiten
                                           IN : IX: Zeiger auf Params
21B6  DD 6E 0D      LD      L,(IX+0D)       lfd. ENV-Zeiger
21B9  DD 66 0E      LD      H,(IX+0E)       nach HL
21BC  DD 5E 10      LD      E,(IX+10)     Schrittzähler
```

```
*********************************** ENV-Gruppe bearbeiten
IN : IX: Zeiger auf Params
                                       HL: Zeiger in ENV-Folge
                                       E: ENV-Schrittzahl
                                   OUT: CY:=1, wenn Kanal deaktiv.
21BF  7B           LD     A,E          Schrittzahl
21C0  FE FF        CP     FF           =$FF b. Tonende?
21C2  28 76        JR     Z,223A       dann Tonende bearbeiten
21C4  87           ADD    A            Schrittzahl b7 gesetzt?
21C5  7E           LD     A,(HL)       Schrittweite laden
21C6  23           INC    HL           Zeiger auf Pausenlänge
21C7  38 4A        JR     C,2213       b7 gesetzt? dann PSG-Hüllkurve
21C9  28 0D        JR     Z,21D8       Null? dann einfach weiter
21CB  1D           DEC    E            Schrittzähler erniedrigen
21CC  B7           OR     A            Schrittweite
21CD  20 06        JR     NZ,21D5      <>0? dann weiter
21CF  DD B6 0F     OR     (IX+0F)      sonst lfd. Lautstärke laden
21D2  F2 DD 21     JP     P,21DD       wenn b7=0, dann weiter
21D5  DD 86 0F     ADD    (IX+0F)      sonst Schrittweite+Lautstärke
21D8  E6 0F        AND    0F           als neue Lautstärke
21DA  CD 73 22     CALL   2273         setzen
21DD  4E           LD     C,(HL)       Pausenlänge laden
21DE  DD 7E 09     LD     A,(IX+09)    Tonlänge Hi laden
21E1  47           LD     B,A          und nach B retten
21E2  87           ADD    A            Tonlänge negativ?
21E3  38 1B        JR     C,2200       dann ggf. nächste ENV-Gruppe
21E5  AF           XOR    A              Zweierkomplement
21E6  91           SUB    C              der Pausenzeit
21E7  DD 86 08     ADD    (IX+08)        zur
21EA  38 0C        JR     C,21F8         laufenden Tonlänge
21EC  05           DEC    B              addieren
21ED  F2 F5 21     JP     P,21F5       noch nicht Tonende? d. weiter
21F0  DD 4E 08     LD     C,(IX+08)    restl. Tonlänge als Pausenzeit
21F3  AF           XOR    A              Tonlänge
21F4  47           LD     B,A            löschen
21F5  DD 70 09     LD     (IX+09),B      restl. Tonlänge
21F8  DD 77 08     LD     (IX+08),A      wieder setzen
21FB  B0           OR     B            Tonende erreicht?
21FC  20 02        JR     NZ,2200      sonst weiter
21FE  1E FF        LD     E,FF         Flag f. Tonende erreicht
2200  7B           LD     A,E          Schrittzahl
2201  B7           OR     A            testen
2202  CC 46 22     CALL   Z,2246       =0? d. nächste ENV-Gruppe
2205  DD 73 10     LD     (IX+10),E    neue Schrittzahl setzen
2208  F3           DI
2209  DD 71 06     LD     (IX+06),C    Pausenzeit setzen
220C  DD 36 07 80  LD     (IX+07),80   Flag f. ENV-Folge aktiv setzen
2210  FB           EI
2211  B7           OR     A            CY:=0
2212  C9           RET

*********************************** PSG-Hüllkurve setzen
                                   IN : E: No. d. PSG-Hüllkurve
                                        A: Lo-Byte ENV-Periode
                                        (HL): Hi-Byte ENV-Periode
2213  57           LD     D,A          Lo-Byte d. Hüllkurvendauer
2214  4B           LD     C,E          Hüllkurvennummer im PSG
2215  3E 0D        LD     A,0D         Nummer d. PSG-Reg. f. ENV-No.
```

```
2217  CD 26 08     CALL   0826         Hüllkurvennummer setzen
221A  4A           LD     C,D          Lo-Byte d. ENV-Periode
221B  3E 0B        LD     A,0B         in den PSG
221D  CD 26 08     CALL   0826         setzen
2220  4E           LD     C,(HL)       Hi-Byte d. ENV-Periode
2221  3E 0C        LD     A,0C         in den PSG
2223  CD 26 08     CALL   0826         setzen
2226  3E 10        LD     A,10         b4:=1 f. Hüllkurvengener. an
2228  CD 73 22     CALL   2273         setzen
222B  CD 46 22     CALL   2246         nächste ENV-Gruppe setzen
222E  7B           LD     A,E          Schrittzahl
222F  3C           INC    A            =$FF?
2230  20 8D        JR     NZ,21BF      sonst nächste ENV-Gr. bearb.
2232  21 B2 21     LD     HL,21B2      dann Default ENV-Kurve
2235  CD 65 22     CALL   2265         einschalten
2238  18 85        JR     21BF         und ENV-Gruppe bearbeiten

***********************************  Kanal deaktivieren
                                     IN : IX: Zeiger auf Params
223A  AF           XOR    A
223B  DD 77 03     LD     (IX+03),A      Status löschen
223E  DD 77 07     LD     (IX+07),A      ENT-Folge inaktiv
2241  DD 77 04     LD     (IX+04),A      ENV-Folge inaktiv
2244  37           SCF                   CY:=1 f. Kanal deaktiviert
2245  C9           RET

***********************************  nächste ENV-Gruppe setzen
                                     IN : IX: Zeiger auf Params
                                          HL: Zeiger auf ENV-Gruppe
                                     OUT: E: Schrittzahl
2246  DD 35 0C     DEC    (IX+0C)        Anz. d. ENV-Gruppen
2249  20 1E        JR     NZ,2269        noch nicht null? d. setzen
224B  DD 7E 09     LD     A,(IX+09)      restl. Tonlänge, Hi
224E  87           ADD    A              Tondauer negativ?
224F  21 B2 21     LD     HL,21B2        sonst Default-Kurve
2252  30 11        JR     NC,2265        setzen
2254  DD 34 08     INC    (IX+08)          Zähler f.
2257  20 06        JR     NZ,225F          Kurvenwiederholungen
2259  DD 34 09     INC    (IX+09)          erhöhen
225C  1E FF        LD     E,FF           Flag f. Tonende
225E  C8           RET    Z              falls Zähler =0

***********************************  lfd. ENV-Kurve initialisieren
                                     IN : IX: Zeigerauf Params
                                     OUT: E: Schrittzahl
225F  DD 6E 0A     LD     L,(IX+0A)        Start d. ENV-Kurve
2262  DD 66 0B     LD     H,(IX+0B)        nach HL laden

***********************************  ENV-Kurve initialisieren
                                     IN : IX: Zeiger auf Params
                                          HL: Zeiger auf Kurvenanfang
                                     OUT: E: Schrittzahl
2265  7E           LD     A,(HL)         Länge der Kurve
2266  DD 77 0C     LD     (IX+0C),A      setzen
2269  23           INC    HL             1. Schrittzahl
226A  5E           LD     E,(HL)         laden
226B  23           INC    HL             Zeiger auf Schrittweite
226C  DD 75 0D     LD     (IX+0D),L        in die Params
```

```
226F  DD 74 0E    LD     (IX+0E),H       speichern
2272  C9          RET

**********************************     Lautstärke setzen
                                       IN : IX: Zeiger auf Params
                                            A: Lautstärke
2273  DD 77 0F    LD     (IX+0F),A       Lautstärke in Params
2276  4F          LD     C,A             und in C
2277  DD 7E 00    LD     A,(IX+00)       Kanalnummer (0-2)
227A  C6 08       ADD    08              ergibt entspr. Lautstärkereg.
227C  C3 26 08    JP     0826            Lautstärke in PSG setzen

**********************************     Kanal ausschalten, aus Aktivität
                                       IN : IX: Zeiger auf Params
227F  DD 7E 01    LD     A,(IX+01)       Kanalmaske
2282  2F          CPL                    invertieren
2283  21 52 B5    LD     HL,B552         Zeiger auf lfd. Aktivitäten
2286  F3          DI
2287  A6          AND    (HL)            aus lfd. Aktivitäten löschen
2288  77          LD     (HL),A          und lfd. Aktivitäten neu setz.
2289  FB          EI
228A  AF          XOR    A               Maske f. Kanal abschalten

**********************************     Kanal an/aus, Rauschen an/aus
                                       IN : IX: Zeiger auf Params
                                            A: Maske f. an/aus
228B  47          LD     B,A             Maske retten
228C  DD 7E 01    LD     A,(IX+01)       Kanalmaske
228F  DD B6 02    OR     (IX+02)         und Rauschmaske
2292  21 19 B6    LD     HL,B619         lfd. PSG-Kontrollbyte
2295  F3          DI
2296  B6          OR     (HL)            mit Kanalmaske verknüpfen
2297  A8          XOR    B               und Maskenbits invertieren
2298  BE          CP     (HL)            sind Änderungen passiert?
2299  77          LD     (HL),A          neues Kontrollbyte setzen
229A  FB          EI
229B  20 03       JR     NZ,22A0         Änderungen? d. entspr. bearb.
229D  78          LD     A,B             Maske
229E  B7          OR     A               Maskenbits gesetzt?
229F  C0          RET    NZ              dann raus
22A0  AF          XOR    A
22A1  CD 76 22    CALL   2276            Kanal ausschalten
22A4  F3          DI
22A5  4E          LD     C,(HL)          neues Kontrollbyte
22A6  3E 07       LD     A,07            ins PSG-Kontrollregister
22A8  C3 26 08    JP     0826            schreiben

**********************************     ENT-Ende bearbeiten
                                       IN : IX: Zeiger auf Params
                                          D,C: lfd. Periodendauer
                                            E: ENT-Folgenummer
                                       OUT: E: Schrittzahl f. ENT
                                          (HL): Zeiger in Folge
22AB  CD 24 23    CALL   2324            Periodendauer setzen
22AE  7B          LD     A,E             ENT-Folgenummer nach A
22AF  CD 4E 23    CALL   234E            Adr. d. ENT-Folge holen
22B2  D0          RET    NC              Fehler? d. raus
22B3  7E          LD     A,(HL)          Repeat-Flag
```

```
22B4  E6 7F       AND   7F            testen
22B6  C8          RET   Z             keine Folgenwiederholung?
22B7  DD 75 11    LD    (IX+11),L       Adr. d. Folge in Params
22BA  DD 74 12    LD    (IX+12),H       schreiben
22BD  CD 13 23    CALL  2313          Schrittzahl holen
22C0  18 09       JR    22CB          und 1. ENT-Gruppe bearbeiten

********************************** lfd. ENT-Gruppe bearbeiten
                                   IN : IX: Zeiger auf Params
22C2  DD 6E 14    LD    L,(IX+14)       lfd. ENT-Zeiger
22C5  DD 66 15    LD    H,(IX+15)       nach HL laden
22C8  DD 5E 18    LD    E,(IX+18)     lfd. Schrittzähler laden

********************************** ENT-Gruppe bearbeiten
                                   IN : IX: Zeiger auf Params
                                        HL: Zeiger in Folge
                                        E: Schrittzähler
22CB  4E          LD    C,(HL)        Schrittweite
22CC  23          INC   HL            Zeiger auf Pausenzeit
22CD  7B          LD    A,E           Schrittzähler
22CE  D6 F0       SUB   F0            <$F0? sonst Periode in A,C
22D0  38 04       JR    C,22D6        dann Schrittweite relativ
22D2  1E 00       LD    E,00          Schrittzahl:=0
22D4  18 0E       JR    22E4          absol. Periode in A,C setzen
22D6  1D          DEC   E             Schrittzähler erniedr.
22D7  79          LD    A,C             Schrittweite in C
22D8  87          ADD   A               vorzeichenerweitert
22D9  9F          SBC   A               nach DC
22DA  57          LD    D,A
22DB  DD 7E 16    LD    A,(IX+16)       und zu Periodendauer
22DE  81          ADD   C               in Params addieren
22DF  4F          LD    C,A             Ergebnis in D,C
22E0  DD 7E 17    LD    A,(IX+17)
22E3  8A          ADC   D
22E4  57          LD    D,A
22E5  CD 24 23    CALL  2324          Periodendauer setzen
22E8  4E          LD    C,(HL)        Pausenzeit laden
22E9  7B          LD    A,E           Schrittzahl
22EA  B7          OR    A             <>0?
22EB  20 19       JR    NZ,2306       d. Zähler u. Flag setzen, raus
22ED  DD 7E 13    LD    A,(IX+13)     sonst Anz d. ENT-Gruppen
22F0  3D          DEC   A             erniedrigen
22F1  20 10       JR    NZ,2303       <>0? d. nächste Gruppe setzen
22F3  DD 6E 11    LD    L,(IX+11)       sonst Adr. d.
22F6  DD 66 12    LD    H,(IX+12)       ENT-Folge laden
22F9  7E          LD    A,(HL)        Wiederholungs-Flag
22FA  C6 80       ADD   80            gesetzt?
22FC  38 05       JR    C,2303        d. 1. ENT-Gruppe setzen
22FE  DD 36 04 00 LD    (IX+04),00    sonst Flag f. ENT-Aktivitüt
2302  C9          RET                   beendet

********************************** nächste ENT-Gruppe aktivieren
                                   IN : IX: Zeiger auf Params
                                        C: Pausenzeit
                                        A: Anz. d. ENT-Gruppen
2303  CD 13 23    CALL  2313          nächste ENT-Gruppe setzen
2306  DD 73 18    LD    (IX+18),E     Schrittzahl setzen
2309  F3          DI
```

```
230A  DD 71 05     LD     (IX+05),C     Pausenlänge setzen
230D  DD 36 04 80  LD     (IX+04),80    Flag f. ENT aktiv setzen
2311  FB           EI
2312  C9           RET

***********************************    nächste ENT-Gruppe setzen
                                       IN : IX: Zeiger Params
                                            A: Anz. d. ENT-Gruppen
                                            HL: Zeiger vor nächste Grup.
                                       OUT: HL: Zeiger in nächste Gruppe
                                            E,A: Schrittzahl
                                             Z:=1, wenn Schrittzahl=0
                                                 , dann E:=01
2313  DD 77 13     LD     (IX+13),A      Anz. d. ENT-Gruppen
2316  23           INC    HL            Schrittzahl
2317  5E           LD     E,(HL)        laden
2318  23           INC    HL            Zeiger in Gruppe
2319  DD 75 14     LD     (IX+14),L       in Parameter
231C  DD 74 15     LD     (IX+15),H       speichern
231F  7B           LD     A,E           Schrittzahl
2320  B7           OR     A             <>0?
2321  C0           RET    NZ            dann zurück
2322  1C           INC    E             sonst Schrittzahl auf 1 setzen
2323  C9           RET

***********************************    Periodendauer setzen
                                       IN : IX: Zeiger auf Params
                                            D,C: Periodendauer
2324  DD 7E 00     LD     A,(IX+00)     Kanalnummer
2327  87           ADD    A             *2=PSG-Register f. Periode, Lo
2328  F5           PUSH   AF            Registernummer retten
2329  DD 71 16     LD     (IX+16),C     Periodendauer, Lo in Params
232C  CD 26 08     CALL   0826          und in PSG-Register schreiben
232F  F1           POP    AF            PSG-Registernummer
2330  3C           INC    A             +1, ergibt Register f. Hi-Byte
2331  4A           LD     C,D           Periodendauer, Hi
2332  DD 71 17     LD     (IX+17),C     in Params
2335  C3 26 08     JP     0826          und in PSG-Register schreiben

***********************************    SOUND AMPL ENVELOPE
                                       IN : A: Nummer d. Hüllkurve
                                            (HL): ENV-Folge, 16 Bytes
2338  11 0A B6     LD     DE,B60A       Start d. ENV-Folgen -$10
233B  18 03        JR     2340          Hüllkurve kopieren

***********************************    SOUND TONE ENVELOPE
                                       IN : A: Nummer d. Hüllkurve
                                            (HL): ENT-Folge, 16 Bytes
233D  11 FA B6     LD     DE,B6FA       Start d. ENT-Folgen -$10

***********************************    Hüllkurve kopieren
                                       IN: A: Nummer d. Hüllkurve
                                           HL: Quelladr. d. Kurve
                                           DE: Beginn d. entspr. Tab.-$10
2340  EB           EX     DE,HL         Tabellenanfang -$10 nach HL
2341  CD 51 23     CALL   2351          Zieladr. f. Kurve berechnen
2344  EB           EX     DE,HL         Quelle nach HL, Ziel nach DE
2345  D0           RET    NC            zurück, wenn Nummer ungültig
```

```
2346  ED B0        LDIR                  Kurve kopieren
2348  C9           RET

***********************************  SOUND A ADDRESS
                                      IN : A: Nummer d. Kurve
                                      OUT: HL: Adresse d. Kurve
                                           CY:=1, wenn gültig
2349  21 0A B6     LD      HL,B60A       Start d. ENV-Folgen -$10
234C  18 03        JR      2351          Adresse berechnen

***********************************  SOUND T ADDRESS
                                      IN : A: Nummer d. Kurve
                                      OUT: HL: Adresse d. Kurve
                                           CY:=1, wenn gültig
234E  21 FA B6     LD      HL,B6FA       Start d. ENT-Folgen -$10

***********************************  Adresse d. Hüllkurve berechnen
                                      IN : A: Nummer d. Kurve
                                           HL: Start d. entspr. Tabelle
                                      OUT: HL: Adresse d. Kurve
                                           CY:=1, wenn gültig
2351  B7           OR      A             Nummer der Kurve
2352  C8           RET     Z             =0? dann Fehler, raus
2353  FE 10        CP      10            >$0F?
2355  D0           RET     NC            dann Fehler, raus
2356  01 10 00     LD      BC,0010       Länge einer Folge
2359  87           ADD     A               Nummer mit 16
235A  87           ADD     A               (f. 16 Bytes/Folge)
235B  87           ADD     A               multiplizieren
235C  87           ADD     A
235D  85           ADD     L               und zum Tabellenstart
235E  6F           LD      L,A             -$10
235F  8C           ADC     H               addieren, ergibt
2360  95           SUB     L               die Adr. d. Kurve
2361  67           LD      H,A             in HL
2362  37           SCF                   CY:=1 f. o.k.
2363  C9           RET
2364  C7           RST     00
2365  C7           RST     00
2366  C7           RST     00
2367  C7           RST     00
2368  C7           RST     00
2369  C7           RST     00
236A  C7           RST     00
236B  C7           RST     00
236C  C7           RST     00
236D  C7           RST     00
236E  C7           RST     00
236F  C7           RST     00
```

```
---------------------- CASSETTE MANAGER (CAS) ----------------------

*********************************  CAS INITIALIZE
2370  CD 01 24     CALL   2401         Eingabe abbrechen
2373  CD 2E 24     CALL   242E         Ausgabe abbrechen
2376  AF           XOR    A            Flag für Meldungen ausgeben
2377  CD 8E 23     CALL   238E         setzen
237A  21 4D 01     LD     HL,014D        Default-SPEED WRITE-
237D  3E 19        LD     A,19           Werte

*********************************  CAS SET SPEED
                                    IN : HL: Hauptzählwert; A: Korrekturwert
237F  29           ADD    HL,HL
2380  29           ADD    HL,HL
2381  29           ADD    HL,HL          b2 bis b9 von Hauptwert
2382  29           ADD    HL,HL          nach H
2383  29           ADD    HL,HL
2384  29           ADD    HL,HL
2385  0F           RRCA                  b2 bis b7 von Korrekturwert
2386  0F           RRCA
2387  E6 3F        AND    3F
2389  6F           LD     L,A            nach L
238A  22 D1 B8     LD     (B8D1),HL    Werte speichern
238D  C9           RET

*********************************  CAS NOISY
                                    IN : A=0 für Meldungen ausgeben
                                         A<>0 für keine Meldungen
238E  32 00 B8     LD     (B800),A     Meldungs-Flag stezen
2391  C9           RET

*********************************  CAS IN OPEN
                                    IN : HL: Adresse des Filenamens
                                         B: Länge des Filenamens
                                         DE: Zeiger auf Eingabebuffer
                                    OUT: HL: Zg. auf Header d. 1. Bl.
                                         DE: Startadresse des Files
                                         BC: Länge des Files
                                         A: Filetyp
                                         CY=0, Z=1 für Abbruch
                                         CY=0, Z=0, wenn schon offen
2392  DD 21 02 B8  LD     IX,B802      Zeiger auf Eingabeparameter
2396  CD AF 23     CALL   23AF         File öffnen
2399  D0           RET    NC           war File schon offen ?
239A  E5           PUSH   HL           Zeiger auf gesuchten Header
239B  CD 3F 25     CALL   253F         1. Block lesen
239E  ED 5B 1C B8  LD     DE,(B81C)    Startadresse,
23A2  ED 4B 1F B8  LD     BC,(B81F)    Länge
23A6  3A 19 B8     LD     A,(B819)     und Filetyp laden
23A9  E1           POP    HL           Zeiger auf Header
23AA  C9           RET

*********************************  CAS OUT OPEN
                                    IN : HL: Adresse des Filenamens
                                         B: Länge des Filenamens
                                         DE: Zeiger auf Ausgabebuffer
                                    OUT: HL: Zg. auf gener. Header;
                                         CY=0, Z=0, wenn schon offen
```

```
23AB  DD 21 47 B8  LD    IX,B847      Zeiger auf Ausgabeparameter
23AF  DD 7E 00     LD    A,(IX+00)    Filestatus
23B2  B7           OR    A            schon offen ?
23B3  C0           RET   NZ           dann Fehler, zurück
23B4  DD E5        PUSH  IX           Zeiger auf Parameter
23B6  E3           EX    (SP),HL      nach HL, Filenamenadr. retten
23B7  36 01        LD    (HL),01      Kennzeichen f. gerade eröffnet
23B9  23           INC   HL
23BA  73           LD    (HL),E         Adresse des Buffers
23BB  23           INC   HL             speichern
23BC  72           LD    (HL),D
23BD  23           INC   HL
23BE  73           LD    (HL),E
23BF  23           INC   HL             Bufferzeiger
23C0  72           LD    (HL),D         auf Bufferanfang setzen
23C1  23           INC   HL
23C2  EB           EX    DE,HL        Zeiger auf Header-Buffer n. DE
23C3  E1           POP   HL           Zeiger auf Filenamen zurück
23C4  D5           PUSH  DE           Zeiger auf Header-Buffer
23C5  0E 40        LD    C,40         Länge des Headers =64 Bytes
23C7  12           LD    (DE),A         Buffer für Header
23C8  13           INC   DE             löschen
23C9  0D           DEC   C
23CA  20 FB        JR    NZ,23C7
23CC  D1           POP   DE           Zeiger auf Header-Buffer
23CD  D5           PUSH  DE
23CE  78           LD    A,B          Länge des Filenamens
23CF  FE 10        CP    10
23D1  38 02        JR    C,23D5       kleiner als 16 ?
23D3  06 10        LD    B,10         sonst Länge auf 16 begrenzen
23D5  04           INC   B            Ausgleich für Predecrement
23D6  48           LD    C,B          Länge+1
23D7  18 07        JR    23E0
23D9  E7           RST   20           RAM LAM, Byte aus Namen holen
23DA  23           INC   HL
23DB  CD B6 27     CALL  27B6         auf Großschrift forcieren
23DE  12           LD    (DE),A       und in Header-Buffer speichern
23DF  13           INC   DE
23E0  10 F7        DJNZ  23D9         weitere Namensbytes ?
23E2  0D           DEC   C            -1 = Länge des Namens
23E3  28 09        JR    Z,23EE       kein Name vorhanden ?
23E5  1B           DEC   DE           Zeiger auf letztes Zeichen
23E6  1A           LD    A,(DE)       letztes Zeichen des Namens
23E7  EE 20        XOR   20           Space ?
23E9  20 03        JR    NZ,23EE      nein ?
23EB  12           LD    (DE),A       sonst durch 0 f. Ende ersetzen
23EC  18 F4        JR    23E2         Ende weiter prüfen
23EE  E1           POP   HL           Zeiger auf Header-Buffer
23EF  DD 36 15 01  LD    (IX+15),01   Nummer des Blocks =1
23F3  DD 36 17 16  LD    (IX+17),16   Filetyp auf ASCII-Datei setzen
23F7  DD 35 1C     DEC   (IX+1C)      Kennzeichen f. 1. Block setzen
23FA  37           SCF                CY=1 für kein Fehler
23FB  C9           RET

*********************************** CAS IN CLOSE
                                    OUT: DE: Zeiger auf Eingabebuffer
                                             CY=0 für nicht offen
23FC  3A 02 B8     LD    A,(B802)     Eingabefile-Status
```

```
23FF  B7            OR    A              nicht offen ?
2400  C8            RET   Z              dann CY=0, zurück

*********************************** CAS IN ABANDON
                                    OUT: DE: Zeiger auf Eingabebuffer
                                         CY=1, Z=0 (immer)
                                         A=$FF f. alle Files geschl.
2401  21 02 B8      LD    HL,B802        Zeiger auf Eingabestatus
2404  3E 01         LD    A,01           Flag für Eingabe
2406  36 00         LD    (HL),00        Kennzeichen für geschlossen
2408  23            INC   HL
2409  5E            LD    E,(HL)           Adresse des
240A  23            INC   HL               Buffers laden
240B  56            LD    D,(HL)
240C  21 CC B8      LD    HL,B8CC        Zeiger auf Ein-/Ausgabeflag
240F  AE            XOR   (HL)           entspr. Bit invertieren
2410  37            SCF                  CY=1 für o.k.
2411  C0            RET   NZ             weitere Kennz. gesetzt ?
2412  77            LD    (HL),A         Ein-/Ausgabeflag auf inaktiv
2413  9F            SBC   A              Z=0, A=$FF
2414  C9            RET

*********************************** CAS OUT CLOSE
                                    OUT: DE: Adr. des Ausgabebuffers
                                         CY=0, Z=1 für Abbruch
                                         CY=0, Z=0 für nicht offen
2415  3A 47 B8      LD    A,(B847)       Ausgabestatus
2418  FE 04         CP    04             Kennzeichen f. abgebrochen ?
241A  28 12         JR    Z,242E         dann keinen Block speichern
241C  C6 FF         ADD   FF             File nicht offen ?
241E  D0            RET   NC             dann zurück, CY=0, Z=0
241F  21 5D B8      LD    HL,B85D          Kennzeichen für letzten
2422  36 FF         LD    (HL),FF          Block setzen
2424  23            INC   HL
2425  23            INC   HL             Zeiger auf Blocklänge
2426  7E            LD    A,(HL)           Blocklänge
2427  23            INC   HL
2428  B6            OR    (HL)             <>0 ?
2429  37            SCF                  CY=1 für kein Abbruch
242A  C4 14 26      CALL  NZ,2614        dann Block speichern
242D  D0            RET   NC             Abbruch ? dann zurück

*********************************** CAS OUT ABANDON
                                    IN : DE: Adr. des Ausgabebuffers
                                         CY=1, Z=0 (immer)
                                         A=$FF f. alle Files geschl.
242E  21 47 B8      LD    HL,B847        Zeiger auf Ausgabeparameter
2431  3E 02         LD    A,02           Flag für Ausgabe
2433  18 D1         JR    2406

*********************************** CAS IN CHAR
                                    OUT: A: Zeichen
                                         CY=0, Z=0 f. EOF/Statusfehl.
                                         CY=0, Z=1 für Abbruch
2435  E5            PUSH  HL
2436  D5            PUSH  DE
2437  C5            PUSH  BC
2438  06 02         LD    B,02           Status f. zeichenweises File
```

```
243A  CD 8B 24    CALL   248B          Status stezen
243D  20 1A       JR     NZ,2459       Status-Fehler ?
243F  2A 1A B8    LD     HL,(B81A)     restliche Bufferlänge
2442  7C          LD     A,H
2443  B5          OR     L             =0 ?
2444  37          SCF                  CY=1 für kein EOF
2445  CC 3F 25    CALL   Z,253F        dann nächsten Block lesen
2448  30 0F       JR     NC,2459       EOF oder Abbruch ?
244A  2A 1A B8    LD     HL,(B81A)     restliche Bufferlänge
244D  2B          DEC    HL            herunterzählen
244E  22 1A B8    LD     (B81A),HL     und wieder setzen
2451  2A 05 B8    LD     HL,(B805)     Bufferzeiger
2454  E7          RST    20            Byte aus RAM holen
2455  23          INC    HL            Bufferzeiger erhöhen
2456  22 05 B8    LD     (B805),HL     und wieder speichern
2459  18 2C       JR     2487          Register vom Stack zurück

**********************************  CAS OUT CHAR
                                    IN : A: Zeichen
                                    OUT: CY=0, Z=0 für Statusfehler
                                         CY=0, Z=1 für Abbruch
245B  E5          PUSH   HL
245C  D5          PUSH   DE
245D  C5          PUSH   BC
245E  4F          LD     C,A           Zeichen
245F  21 47 B8    LD     HL,B847       Zeiger auf Ausgabestatus
2462  06 02       LD     B,02          Status f. zeichenweises File
2464  CD 8E 24    CALL   248E          Status stezen
2467  20 1E       JR     NZ,2487       Status-Fehler ?
2469  2A 5F B8    LD     HL,(B85F)     Bufferlänge
246C  11 00 08    LD     DE,0800       maximale Länge
246F  ED 52       SBC    HL,DE         subtrahieren
2471  C5          PUSH   BC            Zeichen
2472  D4 14 26    CALL   NC,2614       Buffer voll ? d. Block schr.
2475  C1          POP    BC            Zeichen
2476  30 0F       JR     NC,2487       Abbruch ?
2478  2A 5F B8    LD     HL,(B85F)       Bufferlänge
247B  23          INC    HL              erhöhen
247C  22 5F B8    LD     (B85F),HL
247F  2A 4A B8    LD     HL,(B84A)     Bufferzeiger
2482  71          LD     (HL),C        Zeichen in Buffer speichern
2483  23          INC    HL              Bufferzeiger
2484  22 4A B8    LD     (B84A),HL       erhöhen
2487  C1          POP    BC
2488  D1          POP    DE
2489  E1          POP    HL
248A  C9          RET

**********************************  Eingabestatus setzen
                                    IN : B: neuer Status
                                    OUT: CY=0 (immer)
                                         Z=0 für Status-Fehler
248B  21 02 B8    LD     HL,B802
```

```
*********************************** Status setzen
                                    IN : HL: Zeiger auf Status-Byte
                                    B: neuer Status
                                    OUT: CY=0 (immer)
                                            Z=0 für Status-Fehler
248E  7E          LD     A,(HL)        alter Status
248F  B8          CP     B             = neuer Status ?
2490  C8          RET    Z             dann o.k., fertig
2491  EE 01       XOR    01            File gerade geöffnet ?
2493  C0          RET    NZ            nein ? dann Fehler
2494  70          LD     (HL),B        sonst neuen Status setzen
2495  C9          RET

*********************************** CAS TEST EOF
                                    OUT: CY=0 f. EOF (Ende des Files)
2496  CD 35 24    CALL   2435          CAS IN CHAR, Zeichen holen
2499  D0          RET    NC            EOF ?

*********************************** CAS RETURN
249A  E5          PUSH   HL
249B  2A 1A B8    LD     HL,(B81A)
249E  23          INC    HL               Bufferlänge erhöhen
249F  22 1A B8    LD     (B81A),HL
24A2  2A 05 B8    LD     HL,(B805)        und Bufferzeiger
24A5  2B          DEC    HL               auf voriges Zeichen
24A6  22 05 B8    LD     (B805),HL        setzen
24A9  E1          POP    HL
24AA  C9          RET

*********************************** CAS IN DIRECT
                                    IN : HL: Start-Ladeadresse
                                    OUT: HL: Aufrufadresse
                                            CY=0, Z=0 für Statusfehler
                                            CY=0, Z=1 für Abbruch
24AB  EB          EX     DE,HL         Startadresse nach DE
24AC  06 03       LD     B,03          Status für direktes File
24AE  CD 8B 24    CALL   248B          Status setzen
24B1  C0          RET    NZ            Statusfehler ?
24B2  ED 53 1C B8 LD     (B81C),DE     Startadresse setzen
24B6  CD CF 24    CALL   24CF          Buffer kopieren
24B9  2A 1C B8    LD     HL,(B81C)     Startadresse des Blocks
24BC  ED 5B 1A B8 LD     DE,(B81A)     Blocklänge
24C0  19          ADD    HL,DE         addieren
24C1  22 1C B8    LD     (B81C),HL     gibt Startadr. d. nächsten Bl.
24C4  CD 3F 25    CALL   253F          nächsten Block lesen
24C7  38 F0       JR     C,24B9        kein EOF ? dann weiterlesen
24C9  C8          RET    Z             Abbruch ?
24CA  2A A6 B8    LD     HL,(B8A6)     Aufrufadresse laden
24CD  37          SCF                  CY=0 für kein Fehler
24CE  C9          RET

*********************************** Buffer kopieren
24CF  2A 03 B8    LD     HL,(B803)     Zeiger auf Buffer als Quelle
24D2  ED 5B 1C B8 LD     DE,(B81C)     Startadr. des Blocks als Ziel
24D6  ED 4B 1A B8 LD     BC,(B81A)     Länge des Buffers
24DA  7B          LD     A,E
24DB  95          SUB    L                Buffer-Startadresse
24DC  7A          LD     A,D              größer ?
```

```
24DD  9C           SBC    H
24DE  DA A6 BA     JP     C,BAA6       dann KL LDIR, nach unten kop.
24E1  09           ADD    HL,BC        Länge zu Quelladresse addieren
24E2  2B           DEC    HL           Zeiger auf letztes Byte
24E3  EB           EX     DE,HL
24E4  09           ADD    HL,BC        Länge zu Zieladresse addieren
24E5  2B           DEC    HL           Zeiger auf letztes Byte
24E6  EB           EX     DE,HL
24E7  C3 AC BA     JP     BAAC         KL LDDR, Block n. oben versch.

*********************************   CAS OUT DIRECT
                                    IN : HL: Startadresse
                                         DE: Länge
                                         BC: Aufrufadresse
                                         A:  Filetyp
                                    OUT: CY=0, Z=0 für Statusfehler
                                         CY=0, Z=1 für Abbruch
24EA  E5           PUSH   HL           Start-
24EB  C5           PUSH   BC           und Aufrufadresse retten
24EC  4F           LD     C,A          Filetyp
24ED  21 47 B8     LD     HL,B847      Zeiger auf Ausgabestatus
24F0  06 03        LD     B,03         Status für direktes File
24F2  CD 8E 24     CALL   248E         Status setzen
24F5  79           LD     A,C          Filetyp
24F6  C1           POP    BC           Aufrufadresse
24F7  E1           POP    HL           und Startadresse zurück
24F8  C0           RET    NZ           Statusfehler ?
24F9  32 5E B8     LD     (B85E),A     Filetyp,
24FC  ED 53 64 B8  LD     (B864),DE    File-Länge,
2500  ED 43 66 B8  LD     (B866),BC    Aufrufadresse
2504  22 48 B8     LD     (B848),HL    und Startadresse setzen
2507  ED 53 5F B8  LD     (B85F),DE    restliche Länge als Blocklänge
250B  21 FF F7     LD     HL,F7FF      -$801
250E  19           ADD    HL,DE        zu restlicher Länge addieren
250F  3F           CCF                 nur noch ein Restblock ?
2510  D8           RET    C            dann fertig (Block bei CLOSE!)
2511  21 00 08     LD     HL,0800      max. Länge
2514  22 5F B8     LD     (B85F),HL    als Blocklänge setzen
2517  EB           EX     DE,HL          von File-Länge
2518  ED 52        SBC    HL,DE          subtrahieren
251A  E5           PUSH   HL           restliche File-Länge
251B  2A 48 B8     LD     HL,(B848)    Startadresse
251E  19           ADD    HL,DE        Länge addieren
251F  E5           PUSH   HL           gibt nächste Startadresse
2520  CD 14 26     CALL   2614         Block auf Kassette schreiben
2523  E1           POP    HL           neue Startadresse
2524  D1           POP    DE           und restl. File-Länge zurück
2525  D0           RET    NC           Abbruch ?
2526  18 DC        JR     2504         sonst weiter speichern

*********************************   CAS CATALOG
                                    IN/OUT : DE: Eingabebufferadr.
                                    OUT: CY=0 für Statusfehler
2528  21 02 B8     LD     HL,B802      Zeiger auf Eingabestatus
252B  7E           LD     A,(HL)       Status
252C  B7           OR     A            Eingabefile offen ?
252D  C0           RET    NZ           dann Fehler
252E  36 05        LD     (HL),05      Status für Catalog setzen
```

```
2530  ED 53 03 B8  LD    (B803),DE    Bufferadresse setzen
2534  CD 8E 23     CALL  238E         CAS NOISY, Meldungen ermögl.
2537  CD 44 25     CALL  2544         nächsten Block lesen
253A  38 FB        JR    C,2537       kein Abbruch ? d. weiter lesen
253C  C3 01 24     JP    2401         Eingabe abbrechen

*********************************** Block von Kassette lesen
                                    OUT: CY=0, Z=0 für EOF
                                         CY=0, Z=1 für Abbruch
253F  3A 18 B8     LD    A,(B818)     Flag für letzten Block
2542  B7           OR    A            letzter Block ?
2543  C0           RET   NZ           dann Fehler
2544  01 01 83     LD    BC,8301      Nr. f. Meldung/Flag f. Eingabe
2547  CD 73 26     CALL  2673         Meldung ausgeben, Motor ein
254A  30 5C        JR    NC,25A8      Abbruch ?
254C  21 8C B8     LD    HL,B88C      Zeiger auf Buffer für Header
254F  11 40 00     LD    DE,0040      Länge des Headers
2552  3E 2C        LD    A,2C         Header-Kennzeichen
2554  CD 36 28     CALL  2836         CAS READ, Blockheader lesen
2557  30 4F        JR    NC,25A8      Fehler oder Abbruch ?
2559  CD C5 25     CALL  25C5         Namen und Blocknr. vergleichen
255C  20 57        JR    NZ,25B5      nicht gesuchter Name/Block ?
255E  06 8B        LD    B,8B         Nr. für keine Meldung
2560  38 02        JR    C,2564       Catalog-Status ?
2562  06 89        LD    B,89         sonst Nr. für "Loading"
2564  CD 92 26     CALL  2692         Meld., Namen, "block xx" ausg.
2567  ED 5B 9F B8  LD    DE,(B89F)    Länge des Blocks
256B  2A 1C B8     LD    HL,(B81C)    Ladeadresse des Blocks
256E  3A 02 B8     LD    A,(B802)     Eingabestatus
2571  FE 03        CP    03           Direkt-File ?
2573  28 0E        JR    Z,2583       dann lesen
2575  21 FF F7     LD    HL,F7FF      -$801
2578  19           ADD   HL,DE        zu Länge addieren
2579  3E 04        LD    A,04         Nr. des Fehlers "Read error d"
257B  38 2B        JR    C,25A8       Blocklänge >$800 ? dann Fehler
257D  2A 03 B8     LD    HL,(B803)    Bufferadresse
2580  22 05 B8     LD    (B805),HL    Bufferzeiger auf Bufferstart
2583  3E 16        LD    A,16         Block-Kennzeichen
2585  CD 36 28     CALL  2836         CAS READ, Block lesen
2588  30 1E        JR    NC,25A8      Fehler oder Abbruch ?
258A  21 17 B8     LD    HL,B817      Zeiger auf Block-Nr.
258D  34           INC   (HL)         gesuchte Block-Nr. erhöhen
258E  3A 9D B8     LD    A,(B89D)     geles. Kennz. f. letzten Block
2591  23           INC   HL             in Buffer f. gesuchten
2592  77           LD    (HL),A         Header übertragen
2593  AF           XOR   A              Kennzeichen für 1. Block
2594  32 1E B8     LD    (B81E),A       löschen
2597  2A 9F B8     LD    HL,(B89F)    Länge des Blocks
259A  22 1A B8     LD    (B81A),HL    als Bufferlänge
259D  CD BF 27     CALL  27BF         Catalog-Status holen
25A0  3E 8C        LD    A,8C         Nr. für "Ok"
25A2  CC 0C 27     CALL  Z,270C       Catalog ? dann ausgeben
25A5  37           SCF                CY=1 für o.k.
25A6  18 65        JR    260D         Motor ausschalten

*********************************** Lesefehler auswerten
25A8  B7           OR    A            Fehlernr. (0=Abbruch)
25A9  21 02 B8     LD    HL,B802      Zeiger auf Eingabestatus
```

```
25AC  28 5D       JR    Z,260B       Abbruch ? dann Flag setzen
25AE  06 85       LD    B,85         Nr. für "Read error"
25B0  CD 13 27    CALL  2713         Meldung und Fehlernr. ausgeben
25B3  18 97       JR    254C         nächsten Block lesen

********************************** falschen Block auswerten
25B5  F5          PUSH  AF
25B6  06 88       LD    B,88         Nr. für "Found"
25B8  CD 92 26    CALL  2692         Meld., Namen, "block xx" ausg.
25BB  F1          POP   AF           kein Block übersprungen ?
25BC  30 8E       JR    NC,254C      dann nächsten Block lesen
25BE  06 87       LD    B,87         Nr. für "Rewind tape"
25C0  CD 11 27    CALL  2711         Meldung und CR ausgeben
25C3  18 87       JR    254C         nächsten Block lesen

********************************** Namen und Block vergleichen
                                   OUT: Z=0 für Fehler
                                           CY=0 f. falscher Name/Blk.
                                           CY=1 f. Block übersprungen
                                         Z=1 für o.k.
                                           CY=1 f. Catalog
25C5  CD BF 27    CALL  27BF         Catalog-Status holen
25C8  37          SCF                CY=1 für Catalog
25C9  C8          RET   Z            Catalog ? dann zurück
25CA  3A 1E B8    LD    A,(B81E)     Kennzeichen f. 1. Block
25CD  B7          OR    A
25CE  28 1B       JR    Z,25EB       nicht 1. Block gesucht ?
25D0  3A A3 B8    LD    A,(B8A3)     Kennz. f. 1. gelesenen Block
25D3  2F          CPL
25D4  B7          OR    A            nicht 1. Block gelesen ?
25D5  C0          RET   NZ           dann falscher Block, zurück
25D6  3A 07 B8    LD    A,(B807)     1. Byte des gesuchten Namen
25D9  B7          OR    A            ges. Name vorhanden ?
25DA  C4 F3 25    CALL  NZ,25F3      dann Namen vergleichen
25DD  C0          RET   NZ           Namen ungleich ?
25DE  21 8C B8    LD    HL,B88C      Zeiger auf gelesenen Header
25E1  11 07 B8    LD    DE,B807      Zeiger auf gesuchten Header
25E4  01 40 00    LD    BC,0040      Länge des Headers
25E7  ED B0       LDIR               gelesenen in gesuchten kopier.
25E9  AF          XOR   A            CY=0, Z=1
25EA  C9          RET
25EB  CD F3 25    CALL  25F3         Namen vergleichen
25EE  C0          RET   NZ           Namen ungleich ?
25EF  EB          EX    DE,HL        Zeiger auf Blocknummern
25F0  1A          LD    A,(DE)       gesuchte Blocknr.
25F1  BE          CP    (HL)         mit gelesener vergleichen
25F2  C9          RET                CY=1, wenn Block übersprungen

********************************** Namen vergleichen
                                   OUT: Z=1, wenn Namen gleich
                                          CY=0 (immer)
                                          HL,DE: Zeiger auf Blocknrn.
25F3  21 07 B8    LD    HL,B807      Zeiger auf gesuchten Namen
25F6  11 8C B8    LD    DE,B88C      Zeiger auf gelesenen Namen
25F9  06 10       LD    B,10         max. Länge der Namen
25FB  1A          LD    A,(DE)       Byte aus gelesenem Namen
25FC  CD B6 27    CALL  27B6         auf Großschrift forcieren
25FF  4F          LD    C,A          nach C
```

```
2600  7E          LD    A,(HL)      Byte aus gesuchtem Namen
2601  CD B6 27    CALL  27B6        auf Großschrift forcieren
2604  A9          XOR   C           mit gelesenem Byte vergleichen
2605  C0          RET   NZ          ungleich ?
2606  23          INC   HL            Namenzeiger
2607  13          INC   DE            erhöhen
2608  10 F1       DJNZ  25FB        weitere Namensbytes ?
260A  C9          RET

*********************************** Abbruch behandeln
260B  36 04       LD    (HL),04        Status für Abbruch setzen

*********************************** Motor ausschalten
260D  9F          SBC   A              Z=1, wenn Abbruch
260E  F5          PUSH  AF             Fehlerflags retten
260F  CD 4F 2A    CALL  2A4F           CAS STOP MOTOR, Motor aus
2612  F1          POP   AF             Fehlerflags
2613  C9          RET

*********************************** Block auf Kassette schreiben
                                    OUT: CY=0, Z=1 für Abbruch
2614  01 02 84    LD    BC,8402        Nr. d. Meldung/Flag f. Ausgabe
2617  CD 73 26    CALL  2673           Meldung ausgeben, Motor ein
261A  30 4A       JR    NC,2666        Abbruch ?
261C  06 8A       LD    B,8A           Nr. für "Saving"
261E  11 4C B8    LD    DE,B84C        Zeiger auf Filenamen
2621  CD 95 26    CALL  2695           Meld., Name, "block xxx" ausg.
2624  21 63 B8    LD    HL,B863        Zeiger auf Kennz. für 1. Block
2627  CD 88 26    CALL  2688           ggf. auf Abbruch testen
262A  30 3A       JR    NC,2666        Abbruch ?
262C  2A 48 B8    LD    HL,(B848)      Adresse des Ausgabebuffers
262F  22 4A B8    LD    (B84A),HL      Bufferzeiger auf Bufferstart
2632  22 61 B8    LD    (B861),HL      Bufferadr. als Ladeadresse
2635  E5          PUSH  HL             und als Adr. des Blocks
2636  21 4C B8    LD    HL,B84C        Zeiger auf Header
2639  11 40 00    LD    DE,0040        Länge des Headers
263C  3E 2C       LD    A,2C           Header-Kennzeichen
263E  CD 3F 28    CALL  283F           CAS WRITE, Header schreiben
2641  E1          POP   HL             Adresse des Blocks
2642  30 22       JR    NC,2666        Abbruch oder Fehler ?
2644  ED 5B 5F B8 LD    DE,(B85F)      Länge des Blocks
2648  3E 16       LD    A,16           Block-Kennzeichen
264A  CD 3F 28    CALL  283F           CAS WRITE, Block auf Kassette
264D  21 5D B8    LD    HL,B85D        Zeiger a. Kennz. f. letzt. Bl.
2650  DC 88 26    CALL  C,2688         k. Fehler ? ggf. Abbruch test.
2653  30 11       JR    NC,2666        Fehler oder Abbruch ?
2655  21 00 00    LD    HL,0000        Null
2658  22 5F B8    LD    (B85F),HL      als Länge des Buffers setzen
265B  21 5C B8    LD    HL,B85C        Zeiger auf Block-Nr.
265E  34          INC   (HL)           Block-Nr. erhöhen
265F  AF          XOR   A              Kennzeichen für 1. Block
2660  32 63 B8    LD    (B863),A       löschen
2663  37          SCF                  CY=1 für o.k.
2664  18 A7       JR    260D           Motor ausschalten

*********************************** Fehler/Abbruch bei Ausgabe ausw.
2666  B7          OR    A              Fehlernr. (0=Abbruch)
2667  21 47 B8    LD    HL,B847        Zeiger auf Ausgabestatus
```

```
266A  28 9F      JR    Z,260B       Abbruch ? dann behandeln
266C  06 86      LD    B,86         Nr. für "Write error"
266E  CD 13 27   CALL  2713         Meldung und Fehlernr. ausgeben
2671  18 B9      JR    262C         Block nochmals speichern

***********************************  Meldung ausgeben, Motor ein
                                     IN : B: Nr. der Meldung
                                            C: Flag für Ein-/Ausgabe
                                     OUT: CY=0,Z=1, wenn Abbruch
2673  21 CC B8   LD    HL,B8CC        Zeiger auf Ein-/Ausgabeflag
2676  79         LD    A,C            neues Flag
2677  BE         CP    (HL)           mit altem Flag vergleichen
2678  36 00      LD    (HL),00        Flag f. keine Ein-/Ausgabe
267A  37         SCF                  CY=1 für kein Abbruch
267B  E5         PUSH  HL
267C  C5         PUSH  BC
267D  C4 60 27   CALL  NZ,2760        ggf. Meld. ausg./auf Taste w.
2680  C1         POP   BC
2681  E1         POP   HL
2682  9F         SBC   A              Z=1, wenn Abbruch
2683  D0         RET   NC             Abbruch ?
2684  71         LD    (HL),C         sonst Ein-/Ausgabeflag setzen
2685  C3 4B 2A   JP    2A4B           CAS START MOTOR, Motor ein

***********************************  ggf. auf Abbruch testen
                                     IN : HL: Zeiger auf Flag
                                     OUT: CY=0, Z=1, wenn Abbruch
2688  7E         LD    A,(HL)         Flag laden
2689  B7         OR    A
268A  37         SCF                  CY=1 für kein Abbruch
268B  C8         RET   Z              kein Randblock ?
268C  01 2C 01   LD    BC,012C        Verzögerungszähler
268F  C3 72 2A   JP    2A72           auf ESC prüfen, verzögern

2692  11 8C B8   LD    DE,B88C        Zeiger auf gelesenen Namen

***********************************  Meldung, Name, "block xxx " ausg.
                                     IN : B: Nr. der Meldung
                                            DE: Adresse des Namens
2695  3A 00 B8   LD    A,(B800)       Flag für Meldungen
2698  B7         OR    A              keine Meldungen ausgeben ?
2699  C0         RET   NZ             dann zurück
269A  32 01 B8   LD    (B801),A       Flag für Meld. geteilt löschen
269D  CD 83 27   CALL  2783           Cursor auf 1. Spalte
26A0  CD 26 27   CALL  2726           Meldung ausgeben
26A3  1A         LD    A,(DE)         1. Byte des Namens
26A4  B7         OR    A
26A5  20 0A      JR    NZ,26B1        Name vorhanden ?
26A7  3E 8E      LD    A,8E           Nr. für "Unnamed file    "
26A9  CD 27 27   CALL  2727           Meldung ausgeben
26AC  01 10 00   LD    BC,0010        Offset zu Block-Nr.
26AF  18 2E      JR    26DF           Block-Nr. ausgeben
26B1  CD BF 27   CALL  27BF           Flag für Catalog holen
26B4  01 00 10   LD    BC,1000        max. Länge/Zähler f. Namen
26B7  28 0D      JR    Z,26C6         Catalog ? d. 16 Zeichen ausg.
26B9  6B         LD    L,E              Zeiger auf Namen
26BA  62         LD    H,D              nach HL
26BB  7E         LD    A,(HL)         Byte aus Namen
```

```
26BC  B7          OR    A
26BD  28 04       JR    Z,26C3       Ende ?
26BF  0C          INC   C            Länge erhöhen
26C0  23          INC   HL
26C1  10 F8       DJNZ  26BB         weitere Bytes möglich ?
26C3  78          LD    A,B          restlicher Platz f. Namen
26C4  41          LD    B,C          Länge des Namens
26C5  4F          LD    C,A          restlicher Platz als Offset
26C6  CD 8D 27    CALL  278D         ggf. Cursor auf nächste Zeile
26C9  1A          LD    A,(DE)       Byte aus Namen
26CA  CD B6 27    CALL  27B6         auf Großschrift forcieren
26CD  B7          OR    A
26CE  20 02       JR    NZ,26D2      kein Ende ?
26D0  3E 20       LD    A,20         sonst Space
26D2  C5          PUSH  BC
26D3  D5          PUSH  DE
26D4  CD 34 13    CALL  1334         TXT WR CHAR, direkt ausgeben
26D7  D1          POP   DE
26D8  C1          POP   BC
26D9  13          INC   DE
26DA  10 ED       DJNZ  26C9         weitere Namensbytes ?
26DC  CD 5C 27    CALL  275C         Space ausgeben
26DF  EB          EX    DE,HL          Offset zu Namenzeiger
26E0  09          ADD   HL,BC          addieren (B=0!), gibt
26E1  EB          EX    DE,HL          Zeiger auf Blocknr.
26E2  3E 8D       LD    A,8D         Nr. für "block"
26E4  CD 27 27    CALL  2727         Meldung ausgeben
26E7  06 02       LD    B,02         2 Zeichen für Blocknr.
26E9  CD 8D 27    CALL  278D         ggf. Cursor auf nächste Zeile
26EC  1A          LD    A,(DE)       Nr. des Blocks
26ED  CD A4 27    CALL  27A4         ausgeben
26F0  CD 5C 27    CALL  275C         Space ausgeben
26F3  13          INC   DE           Zeiger auf Endblock-Kennzeich.
26F4  CD BF 27    CALL  27BF         Catalog-Flag holen
26F7  20 0B       JR    NZ,2704      nicht Catalog ?
26F9  13          INC   DE           Zeiger auf Filetyp
26FA  1A          LD    A,(DE)       Filetyp
26FB  E6 0F       AND   0F             entsprechendes ASCII-Kennz.
26FD  C6 24       ADD   24             generieren
26FF  CD 80 27    CALL  2780         ausgeben
2702  18 58       JR    275C         Space ausgeben
2704  1A          LD    A,(DE)       Flag für letzten Block
2705  21 01 B8    LD    HL,B801      Flag f. Meldung geteilt
2708  B6          OR    (HL)         letzter Bl. o. Meld. geteilt ?
2709  C8          RET   Z            nein ? dann zurück
270A  18 6F       JR    277B         CR ausgeben (f. nächste Meld.)

*********************************** Meldung und CR ausgeben
                                    IN : A: Nr. der Meldung
270C  CD 27 27    CALL  2727           Meldung ausgeben
270F  18 6A       JR    277B           CR ausgeben

*********************************** Meldung am linken Rand & CR ausg.
                                    IN : B: Nr. der Meldung
2711  3E FF       LD    A,FF           Flag für folgendes CR
```

```
*********************************** Meldung und Fehlernr. ausgeben
                                    IN : A: Nr. des Fehlers
                                           A=$FF für CR
                                           B: Nr. der Meldung
2713  F5         PUSH   AF            Fehlernr./Flag retten
2714  CD 1F 27   CALL   271F          Meldung am linken Rand ausg.
2717  F1         POP    AF            Fehlernr./Flag
2718  C6 60      ADD    60            ASCII-Code des Fehlers gener.
271A  D4 80 27   CALL   NC,2780       nicht zu groß ? dann ausgeben
271D  18 5C      JR     277B          CR ausgeben

*********************************** Meldung am linken Rand ausgeben
                                    IN : B: Nr. der Meldung
271F  CD 80 11   CALL   1180          Cursorposition holen
2722  25         DEC    H             Spalte=1 ?
2723  C4 7B 27   CALL   NZ,277B       nein ? dann CR ausgeben

*********************************** Meldung ausgeben
                                    IN : B: Nr. der Meldung
2726  78         LD     A,B           Nr. der Meldung
2727  E5         PUSH   HL            (Zeiger in Meldung retten)
2728  E6 7F      AND    7F            Untermeldungs-Kennz. löschen
272A  47         LD     B,A           Nr. der Meldung
272B  21 C5 27   LD     HL,27C5       Zeiger auf Start der Meldungen
272E  28 07      JR     Z,2737        Meldungs-Nr. =0 ? dann ausg.
2730  7E         LD     A,(HL)        Zeichen aus Tabelle
2731  23         INC    HL
2732  B7         OR     A             nicht Ende der Meldung ?
2733  20 FB      JR     NZ,2730       dann weiter suchen
2735  10 F9      DJNZ   2730          weitere Meldungen übergehen ?
2737  7E         LD     A,(HL)        Zeichen aus Meldung
2738  B7         OR     A
2739  28 05      JR     Z,2740        Ende ?
273B  CD 43 27   CALL   2743          Wort und Space ausgeben
273E  18 F7      JR     2737          nächstes Wort
2740  E1         POP    HL            Zg. in Meldung (bei Rekursion)
2741  23         INC    HL            auf Zeichen nach Meldungs-Nr.
2742  C9         RET

*********************************** Wort und Space ausgeben
                                    IN : A: 1. Zeichen
                                            S: b7 des 1. Zeichen
                                    IN/OUT: HL: Meldungs-Zeiger
2743  FA 27 27   JP     M,2727        b7 gesetzt ? d. Meldung ausg.
2746  E5         PUSH   HL            Zeiger in Meldung
2747  06 00      LD     B,00          Zähler f. Wortlänge
2749  04         INC    B             erhöhen
274A  7E         LD     A,(HL)        Zeichen aus Wort
274B  23         INC    HL
274C  07         RLCA
274D  30 FA      JR     NC,2749       kein Wortende ?
274F  CD 8D 27   CALL   278D          ggf. Cursor auf nächste Zeile
2752  E1         POP    HL            Zeiger in Meldung auf Wort
2753  7E         LD     A,(HL)        Zeichen aus Wort
2754  23         INC    HL
2755  E6 7F      AND    7F            Endkennz. löschen
2757  CD 80 27   CALL   2780          Zeichen ausgeben
275A  10 F7      DJNZ   2753          weitere Zeichen im Wort ?
```

```
275C  3E 20      LD    A,20        Space
275E  18 20      JR    2780        ausgeben

*********************************  Meldung ausgeben, a. Taste warten
                                   IN : B: Nr. der Meldung
                                   OUT: CY=0, Z=1 für Abbruch
2760  3A 00 B8   LD    A,(B800)      Flag für Meldungen
2763  B7         OR    A             keine Meldungen ausgeben ?
2764  37         SCF                 CY=1 für kein Abbruch
2765  C0         RET   NZ            dann zurück
2766  CD 1F 27   CALL  271F          Meldung am linken Rand ausg.
2769  CD 42 1A   CALL  1A42          KM READ CHAR, Taste lesen
276C  38 FB      JR    C,2769        bis keine Taste mehr im Buffer
276E  CD 79 12   CALL  1279          TXT CUR ON, Cursor einschalten
2771  CD 56 1B   CALL  1B56          KM WAIT KEY, auf Taste warten
2774  CD 81 12   CALL  1281          TXT CUR OFF, Cursor wieder aus
2777  FE 1B      CP    1B            CTRL-eckige Klammer auf ? (??)
2779  C8         RET   Z             dann zurück
277A  37         SCF                 CY=1 für kein Abbruch
277B  CD 83 27   CALL  2783          Cursor auf 1. Spalte
277E  3E 0A      LD    A,0A          Code für Linefeed
2780  C3 00 14   JP    1400          Zeichen ausgeben

*********************************  Cursor auf 1. Spalte setzen
2783  F5         PUSH  AF
2784  E5         PUSH  HL
2785  3E 01      LD    A,01          Spalte=1
2787  CD 5E 11   CALL  115E          TXT SET COLUMN, Spalte setzen
278A  E1         POP   HL
278B  F1         POP   AF
278C  C9         RET

*********************************  ggf. Cursor auf nächste Zeile
                                   IN : B: Wortlänge
                                   OUT: CY=0, wenn Wort zu lang
278D  D5         PUSH  DE
278E  CD 56 12   CALL  1256          TXT GET WINDOW, Grenzen holen
2791  5C         LD    E,H           linke Windowgrenze
2792  CD 80 11   CALL  1180          TXT GET CURSOR, Cursorpos. h.
2795  7C         LD    A,H           Cursorspalte
2796  3D         DEC   A             -1 f. absolute Koordinaten
2797  83         ADD   E             + linke Windowgrenze
2798  80         ADD   B             + Wortlänge
2799  3D         DEC   A             -1 f. absolute Koordinaten
279A  BA         CP    D             mit rechter Winowgrenze vergl.
279B  D1         POP   DE
279C  D8         RET   C             Wort innerhalb der Zeile ?
279D  3E FF      LD    A,FF          Flag für Meldung geteilt
279F  32 01 B8   LD    (B801),A      setzen
27A2  18 D7      JR    277B          CR ausgeben

*********************************  Dezimalzahl ausgeben
                                   IN : A: Zahl
27A4  06 FF      LD    B,FF          Zähler für Zehnerstelle
27A6  04         INC   B             Zehnerstelle erhöhen
27A7  D6 0A      SUB   0A            10 von Einerstelle abziehen
27A9  30 FB      JR    NC,27A6       ggf. weiter abziehen
27AB  C6 3A      ADD   3A            "0"+10 addieren, ASCII-Code
```

```
27AD  F5            PUSH   AF          Code retten
27AE  78            LD     A,B         Zehnerstelle
27AF  B7            OR     A           <>0 ?
27B0  C4 A4 27      CALL   NZ,27A4     dann ausgeben
27B3  F1            POP    AF          Einerstelle
27B4  18 CA         JR     2780        ausgeben

**********************************  auf Großschrift forcieren
                                    IN/OUT: A: Zeichen
27B6  FE 61         CP     61          < "a" ?
27B8  D8            RET    C           dann zurück
27B9  FE 7B         CP     7B          > "z"+1 ?
27BB  D0            RET    NC          dann zurück
27BC  C6 E0         ADD    E0          sonst nach Großschrift wandeln
27BE  C9            RET

**********************************  Catalog-Flag holen
                                    OUT: Z=1 für Catalog
27BF  3A 02 B8      LD     A,(B802)    Eingabestatus
27C2  FE 05         CP     05          Status für Catalog ?
27C4  C9            RET

**********************************  Kassetten-Meldungen
27C5  50 72 65 73 F3 00                00 "Press "
27CB  50 4C 41 D9 74 68 65 EE          01 "PLAY then any key: "
27D3  61 6E F9 6B 65 79 BA 00
27DB  65 72 72 6F F2 00                02 "error "
27E1  80 81 00                         03 "Press PLAY then any key: "
27E4  80 52 45 C3 61 6E E4             04 "Press REC and "
27EB  81 00                               "PLAY then any key: "
27ED  52 65 61 E4 82 00                05 "Read error "
27F3  57 72 69 74 E5 82 00             06 "Write error "
27FA  52 65 77 69 6E E4 74 61          07 "Rewind tape "
2802  70 E5 00
2805  46 6F 75 6E 64 20 A0 00          08 "Found   "
280D  4C 6F 61 64 69 6E E7 00          09 "Loading "
2815  53 61 76 69 6E E7 00             0A "Saving "
281C  00                               0B ""
281D  4F EB 00                         0C "Ok "
2820  62 6C 6F 63 EB 00                0D "block "
2826  55 6E 6E 61 6D 65 E4 66          0E "Unnamed file     "
282E  69 6C 65 20 20 20 A0 00

**********************************  CAS READ
                                    IN : HL: Ladeadresse
                                         DE: Länge
                                         A: Block-Kennzeichen
                                           $2C für Header
                                           $16 für Datenblock
                                    OUT: CY=0 für Fehler/Abbruch
                                         A: Fehlernr. (0=Abbruch)
2836  CD 73 28      CALL   2873        Motor an, Tastatur vorbereiten
2839  F5            PUSH   AF          alten Motor-Status retten
283A  21 B8 28      LD     HL,28B8     Routine für eine Page lesen
283D  18 19         JR     2858
```

```
*********************************  CAS WRITE
                                   IN : HL: Startadresse
                                        DE: Länge
                                        A: Block-Kennzeichen
                                          $2C für Header
                                          $16 für Datenblock
                                   OUT: CY=0 für Fehler/Abbruch
A: Fehlernr. (0=Abbruch)
283F  CD 73 28    CALL   2873         Motor an, Tastatur vorbereiten
2842  F5          PUSH   AF           alten Motor-Status retten
2843  CD 64 29    CALL   2964         Synchronisationston schreiben
2846  21 F7 28    LD     HL,28F7      Routine f. eine Page schreiben
2849  DC 9D 28    CALL   C,289D       o.k. ? dann Bereich schreiben
284C  DC 79 29    CALL   C,2979       o.k. ? dann End-Ton schreiben
284F  18 0F       JR     2860         Motor und Tastatur wieder zur.

*********************************  CAS CHECK
                                   IN : HL: Startadresse
                                        DE: Länge
                                        A: Block-Kennzeichen
                                          $2C für Header
                                          $16 für Datenblock
                                   OUT: CY=0 für Fehler/Abbruch
                                        A: Fehlernr. (0=Abbruch)
2851  CD 73 28    CALL   2873         Motor an, Tastatur vorbereiten
2854  F5          PUSH   AF           alten Motor-Status retten
2855  21 C7 28    LD     HL,28C7      Rout. f. eine Page vergleichen
2858  E5          PUSH   HL           Routinenadresse retten
2859  CD 19 29    CALL   2919         auf Synchronisation warten
285C  E1          POP    HL           Routinenadresse zurück
285D  DC 9D 28    CALL   C,289D       o.k. ? dann lesen/vergleichen
2860  D1          POP    DE           alter Motor-Status
2861  F5          PUSH   AF           Fehlerflag und Nr. retten
2862  01 82 F7    LD     BC,F782      PIO, Steuerregister
2865  ED 49       OUT    (C),C        Port A wieder auf Ausgabe
2867  01 10 F6    LD     BC,F610      PIO, Port C
286A  ED 49       OUT    (C),C        PSG inaktiv, Tastaturzeile 0
286C  FB          EI
286D  7A          LD     A,D          alter Motor-Status
286E  CD 51 2A    CALL   2A51         Motor wieder enstprechend
2871  F1          POP    AF           Fehlerflag und Nr.
2872  C9          RET

*********************************  Motor ein, Tastatur vorbereiten
                                   IN : HL: Startadresse
                                        DE: Länge
                                        A: Block-Kennzeichen
                                   OUT: IX: Startadresse
                                        DE: modifizierte Länge
2873  32 CD B8    LD     (B8CD),A     Block-Kennzeichen setzen
2876  1B          DEC    DE             D:=Zahl der ganzen Pages
2877  1C          INC    E              E:=rest. Bytes (0 f. 1 Page)
2878  E5          PUSH   HL           Startadresse
2879  D5          PUSH   DE           modifizierte Länge
287A  CD 68 1E    CALL   1E68         SOUND RESET
287D  D1          POP    DE           modifizierte Länge
287E  DD E1       POP    IX           Startadresse
2880  CD 4B 2A    CALL   2A4B         CAS START MOTOR, Motor ein
```

```
2883  F3          DI                 Interrupts verhindern
2884  01 0E F4    LD      BC,F40E    PIO, Port A
2887  ED 49       OUT     (C),C      PSG, Registernr. für Port 0
2889  01 D0 F6    LD      BC,F6D0    PIO, Port C
288C  ED 49       OUT     (C),C      Motor ein, PSG Adr. übernehmen
288E  0E 10       LD      C,10
2890  ED 49       OUT     (C),C      Motor ein, PSG inaktiv
2892  01 92 F7    LD      BC,F792    PIO, Steuerregister
2895  ED 49       OUT     (C),C      Port A auf Eingabe
2897  01 58 F6    LD      BC,F658    PIO, Port C
289A  ED 49       OUT     (C),C      Tastaturzl. 8 f. ESC, Read PSG
289C  C9          RET

**********************************  Block pageweise bearbeiten
                                 IN : IX: Block-Start-/Ladeadresse
                                      HL: Routinenadr. f. 1 Page
                                      DE: modifizierte Länge
                                      D: Zahl der ganzen Pages
                                      E: Zahl d. Bytes in Restpage
                                         (0 für eine ganze Page)
                                 OUT: CY=0, wenn Fehler
                                      A: Fehlernr.
289D  7A          LD      A,D          Zahl der ganzen Pages
289E  B7          OR      A            nur 1 Restpage (Länge<=$100) ?
289F  28 0D       JR      Z,28AE       dann letzte Page bearbeiten
28A1  E5          PUSH    HL           Routinenadresse
28A2  D5          PUSH    DE           modifizierte Länge
28A3  1E 00       LD      E,00         Kennz. für Datenlänge =$100
28A5  CD AE 28    CALL    28AE         eine Page bearbeiten
28A8  D1          POP     DE           modifizierte Länge
28A9  E1          POP     HL           Routinenadresse
28AA  D0          RET     NC           Fehler ? dann zurück
28AB  15          DEC     D            Zahl der ganzen Pages
28AC  20 F3       JR      NZ,28A1      weitere Pages ?
28AE  01 FF FF    LD      BC,FFFF        Check-Word
28B1  ED 43 D3 B8 LD      (B8D3),BC      initialisieren
28B5  16 01       LD      D,01         Kennz. f. Blocklänge =$100
28B7  E9          JP      (HL)         Routine anspringen

**********************************  eine Page lesen
                                 IN : IX: Ladeadresse
                                      E: Zahl der Datenbytes
                                         (0 für ganze Page)
                                      D: Zahl d. Bytes insgesamt+1
                                         (1 für ganze Page)
                                 OUT: CY=0, wenn Fehler
                                      A: Fehlernr.
28B8  CD B0 29    CALL    29B0         ein Byte lesen
28BB  D0          RET     NC           Fehler ?
28BC  DD 77 00    LD      (IX+00),A    sonst Byte abspeichern
28BF  DD 23       INC     IX
28C1  15          DEC     D            Länge insgesamt
28C2  1D          DEC     E            Datenlänge
28C3  20 F3       JR      NZ,28B8      weitere Datenbytes ?
28C5  18 12       JR      28D9         Restblock und Blockende lesen
```

```
**********************************   eine Page lesen und vergleichen
                                     IN : IX: Startadresse
                                          E: Zahl der Datenbytes
                                             (0 für ganze Page)
                                          D: Zahl d. Bytes insgesamt+1
                                             (1 für ganze Page)
                                     OUT: CY=0, wenn Fehler
                                          A: Fehlernr.
28C7  CD B0 29    CALL   29B0           ein Byte lesen
28CA  D0          RET    NC             Fehler ?
28CB  47          LD     B,A            gelesenes Byte
28CC  CD DC BA    CALL   BADC           Byte aus RAM holen
28CF  A8          XOR    B              mit gelesenem Byte vergleichen
28D0  3E 03       LD     A,03           Fehlernr. für "Read error c"
28D2  C0          RET    NZ             ungleich ? dann Fehler
28D3  DD 23       INC    IX
28D5  15          DEC    D              Länge insgesamt
28D6  1D          DEC    E              Datenlänge
28D7  20 EE       JR     NZ,28C7        weitere Datenbytes ?
28D9  15          DEC    D              weitere Bytes in Restblock ?
28DA  28 06       JR     Z,28E2         nein ?
28DC  CD B0 29    CALL   29B0           Byte lesen
28DF  D0          RET    NC             Fehler ?
28E0  18 F7       JR     28D9           weitere Füllbytes lesen
28E2  CD A6 29    CALL   29A6           Check-Word holen
28E5  CD B0 29    CALL   29B0           Byte lesen
28E8  D0          RET    NC             Fehler ?
28E9  AA          XOR    D              mit Check-Word hi vergleichen
28EA  20 07       JR     NZ,28F3        ungleich ?
28EC  CD B0 29    CALL   29B0           Byte lesen
28EF  D0          RET    NC             Fehler ?
28F0  AB          XOR    E              mit Check-Word lo vergleichen
28F1  37          SCF                   CY=1 für kein Fehler
28F2  C8          RET    Z              Check-Word o.k. ?
28F3  3E 02       LD     A,02           Nr. für Check-Word-Fehler
28F5  B7          OR     A              CY=0 für Fehler
28F6  C9          RET

**********************************   eine Page auf Band schreiben
                                     IN : IX: Startadresse
                                          E: Zahl der Datenbytes
                                             (0 für ganze Page)
                                          D: Zahl d. Bytes insgesamt+1
                                             (1 für ganze Page)
                                     OUT: CY=0, wenn Fehler
                                          A: Fehlernr.
28F7  CD DC BA    CALL   BADC           Byte aus RAM holen
28FA  CD F8 29    CALL   29F8           Byte speichern
28FD  D0          RET    NC             Fehler ?
28FE  DD 23       INC    IX
2900  15          DEC    D              Länge insgesamt
2901  1D          DEC    E              Daten-Länge
2902  20 F3       JR     NZ,28F7        weitere Datenbytes ?
2904  15          DEC    D              weitere Restblock-Bytes ?
2905  28 07       JR     Z,290E         nein ?
2907  AF          XOR    A              sonst Null
2908  CD F8 29    CALL   29F8           als Füllbyte auf Band
290B  D0          RET    NC             Fehler ?
```

```
2902C  18 F6        JR     2904         weitere Füllbytes
290E   CD A6 29     CALL   29A6         Check-Word holen
2911   CD F8 29     CALL   29F8         Check-Word hi auf Band
2914   D0           RET    NC           Fehler ?
2915   7B           LD     A,E          Check-Word lo
2916   C3 F8 29     JP     29F8         auf Band

********************************** auf Synchronisation warten
                                   OUT: CY=1, Z=1, A=0 für Abbruch
2919   D5           PUSH   DE
291A   CD 23 29     CALL   2923          Sync. lesen, Baudr./Flanke ho.
291D   D1           POP    DE
291E   D8           RET    C            kein Fehler ?
291F   B7           OR     A            Abbruch ?
2920   C8           RET    Z            dann zurück
2921   18 F6        JR     2919         sonst weiter versuchen

********************************** Synchronisation lesen/auswerten
                                   OUT: ($B8CE): Flanken-Flag
                                                 ($55 bzw. $AA)
                                          ($B8CF): Zählengrenzen-Wert
                                                 (abh. v. Baudrate)
2923   2E 55        LD     L,55          Flanken-Byte, wechs. pos./neg.
2925   CD CD 29     CALL   29CD          auf nächste Flanke warten
2928   D0           RET    NC           Fehler ?
2929   11 00 00     LD     DE,0000       Zeitzähler =0
292C   62           LD     H,D           Flankenzähler =0
292D   CD CD 29     CALL   29CD          auf nächste Flanke warten
2930   D0           RET    NC           Fehler ?
2931   EB           EX     DE,HL
2932   06 00        LD     B,00          akt. Zeitzähler hi=0
2934   09           ADD    HL,BC         akt. Zeitzähler addieren
2935   EB           EX     DE,HL
2936   25           DEC    H            Flankenzähler
2937   20 F4        JR     NZ,292D       weitere Flanken ?
2939   61           LD     H,C           vorigen Zeitzähler nach H
293A   79           LD     A,C           Zeitzähler
293B   92           SUB    D            - bish. mittlere Flankenzeit
293C   4F           LD     C,A           als Lo-Byte
293D   9F           SBC    A            Vorzeichen erweitern
293E   47           LD     B,A           Hi-Byte
293F   EB           EX     DE,HL           Differenz zu mittlerer
2940   09           ADD    HL,BC           Flankenzeit addieren
2941   EB           EX     DE,HL           (Gewicht 1:256)
2942   CD CD 29     CALL   29CD          auf nächste Flanke warten
2945   D0           RET    NC           Fehler ?
2946   7A           LD     A,D           mittlere Flankenzeit
2947   CB 3F        SRL    A
2949   CB 3F        SRL    A              mit 5/4 multiplizieren
294B   8A           ADC    D              (1/4 als Toleranzgrenze)
294C   94           SUB    H            vorige Flankenzeit abziehen
294D   38 EA        JR     C,2939        Flankenzeit zu groß ?
294F   91           SUB    C            akt. Flankenzeit abziehen
2950   38 E7        JR     C,2939        zu groß für 2 0-Bit-Flanken ?
2952   7A           LD     A,D            mittl. Flankenzeit f. 1-Bit
2953   1F           RRA                   mal 3/2 als Zeitzählgrenze
2954   8A           ADC    D              (für doppelte Flankenzeit!)
2955   67           LD     H,A            Zeitzählgrenze zusammen mit
```

```
2956  22 CE B8    LD     (B8CE),HL       Flankenflag abspeichern
2959  CD B0 29    CALL   29B0           Byte lesen
295C  D0          RET    NC             Fehler ?
295D  21 CD B8    LD     HL,B8CD        Zeiger auf Kennbyte
2960  AE          XOR    (HL)           Byte mit gesuchtem vergleichen
2961  C0          RET    NZ             ungleich ? dann Fehler
2962  37          SCF                   CY=1 für kein Fehler
2963  C9          RET

**********************************   Synchronisation schreiben
                                     OUT: CY=0 für Fehler/Abbruch
                                          A: Fehlernr. (0=Abbruch)
2964  CD 89 2A    CALL   2A89           Verzögerung
2967  21 01 08    LD     HL,0801        Zahl der 1-Bits
296A  CD 7C 29    CALL   297C           1-Bits als Synchronisation
296D  D0          RET    NC             Abbruch ?
296E  B7          OR     A              CY=0 für 0-Bit
296F  CD 08 2A    CALL   2A08           Bit auf Band
2972  D0          RET    NC             Fehler ?
2973  3A CD B8    LD     A,(B8CD)       Kennbyte
2976  C3 F8 29    JP     29F8           auf Band schreiben

**********************************   Blockendton schreiben
                                     OUT: CY=0, A=0 für Abbruch
2979  21 21 00    LD     HL,0021        Zahl der 1-Bits
297C  06 F4       LD     B,F4           PIO, Port A
297E  ED 78       IN     A,(C)          Tastatur-Rückmeldung Zeile 8
2980  E6 04       AND    04             Bit für ESC isolieren
2982  C8          RET    Z              ESC gedrückt ?
2983  E5          PUSH   HL             Bitzähler retten
2984  37          SCF                   CY=1 für 1-Bit
2985  CD 08 2A    CALL   2A08           Bit auf Band schreiben
2988  E1          POP    HL             Bitzähler
2989  2B          DEC    HL
298A  7C          LD     A,H
298B  B5          OR     L
298C  20 EE       JR     NZ,297C        weitere Bits ?
298E  37          SCF                   CY=1 für kein Abbruch
298F  C9          RET

**********************************   Bit in Check-Word 'reinmurksen
                                     IN : A=$FF, wenn 1-Bit
                                          A=0, wenn 0-Bit
2990  2A D3 B8    LD     HL,(B8D3)      altes Check-Word
2993  AC          XOR    H              mit neuen Bit verknüpfen
2994  F2 A0 29    JP     P,29A0         neues Bit XOR b15 =0 ?
2997  7C          LD     A,H
2998  EE 08       XOR    08               b4 und b11 des
299A  67          LD     H,A              Check-Words invertieren
299B  7D          LD     A,L
299C  EE 10       XOR    10
299E  6F          LD     L,A
299F  37          SCF                   CY=1 für 1-Bit
29A0  ED 6A       ADC    HL,HL          Bit in Check-Word rotieren
29A2  22 D3 B8    LD     (B8D3),HL      Checkword wieder speichern
29A5  C9          RET
```

```
***********************************   Check-Word holen
                                      OUT: DE: Check-Word
                                            A: Hi-Byte
29A6  2A D3 B8    LD     HL,(B8D3)      Check-Word
29A9  7D          LD     A,L
29AA  2F          CPL
29AB  5F          LD     E,A              invertieren, nach DE
29AC  7C          LD     A,H
29AD  2F          CPL
29AE  57          LD     D,A
29AF  C9          RET

***********************************   Byte einlesen
                                      OUT: A: Byte
                                           CY=0 für Fehler
                                           A: Fehlernr.
29B0  D5          PUSH   DE
29B1  1E 08       LD     E,08           Zähler für 8 Bits
29B3  2A CE B8    LD     HL,(B8CE)      Zeitgrenze/Flankenflag
29B6  CD D4 29    CALL   29D4           auf 1. Flanke des Bits warten
29B9  DC DD 29    CALL   C,29DD         o.k. ? d. auf 2. Flanke warten
29BC  30 0D       JR     NC,29CB        Fehler ?
29BE  7C          LD     A,H            Zeitzählergrenze
29BF  91          SUB    C              minus akt. Zeitzähler gibt Bit
29C0  9F          SBC    A              A=$FF bei 1-Bit
29C1  CB 12       RL     D              Bit in Byte rotieren
29C3  CD 90 29    CALL   2990           Check-W. entspr. Bit neu setz.
29C6  1D          DEC    E              Bitzähler
29C7  20 EA       JR     NZ,29B3        weitere Bits ?
29C9  7A          LD     A,D            gelesenes Byte
29CA  37          SCF                   CY=1 für kein Fehler
29CB  D1          POP    DE
29CC  C9          RET

***********************************   auf nächste Flanke warten, ESC t.
                                      IN/OUT: L: Flankenflag
                                                 L=$55 für neg. Flanke
                                                 L=$AA für pos. Flanke
                                                 (OUT für die andere Fl.)
                                      OUT: C: Zeitzähler
                                           CY=0, Z=1, A=0, wenn Abbruch
                                           CY=0, Z=0, wenn Fehler dann: A: Fehlernr.
29CD  06 F4       LD     B,F4           PIO, Port B
29CF  ED 78       IN     A,(C)          Tastatur-Rückmeld. d. 8. Zeile
29D1  E6 04       AND    04             Bit für ESC isolieren
29D3  C8          RET    Z              ESC gedrückt ?

***********************************   auf nächste Flanke warten
                                      IN/OUT: L: Flankenflag
                                                 L=$55 für neg. Flanke
                                                 L=$AA für pos. Flanke
                                                 (OUT für die andere Fl.)
                                      OUT: C: Zeitzähler
                                           CY=0, Z=0, wenn Fehler dann: A: Fehlernr.
29D4  ED 5F       LD     A,R            Zeit seit letzter Flanke
29D6  C6 03       ADD    03
29D8  0F          RRCA                    entspr. Zeitzähler-Wert
29D9  0F          RRCA                    generieren
```

```
29DA  E6 1F        AND    1F
29DC  4F           LD     C,A          als Zeitzähler nach C

**********************************  auf 2. Flanke im Bit warten
                                    IN/OUT: L: Flankenflag
                                               L=$55 für neg. Flanke
                                               L=$AA für pos. Flanke
                                               (OUT für die andere Fl.)
                                               C: Zeitzähler
                                    OUT: CY=0, Z=0, wenn Fehler dann: A: Fehlernr.
29DD  06 F5        LD     B,F5         PIO, Port B
29DF  79           LD     A,C          Zeitzähler
29E0  C6 02        ADD    02           erhöhen
29E2  4F           LD     C,A          wieder setzen
29E3  38 0E        JR     C,29F3       Übertrag ? dann Zeit zu lang
29E5  ED 78        IN     A,(C)        Eingabebit von Kassette laden
29E7  AD           XOR    L            je nach Flankenflag inv.
29E8  E6 80        AND    80           Eingabebit isolieren
29EA  20 F3        JR     NZ,29DF      nicht gesuchter Pegel ?
29EC  AF           XOR    A              Zeitzähler im Refresh-
29ED  ED 4F        LD     R,A            Register rücksetzen
29EF  CB 0D        RRC    L            Flankenflag f. nächste Flanke
29F1  37           SCF                 CY=1 für kein Fehler
29F2  C9           RET
29F3  AF           XOR    A              Zeitzähler im Refresh-
29F4  ED 4F        LD     R,A            Register rücksetzen
29F6  3C           INC    A            Fehlernr. =1, CY=0
29F7  C9           RET

**********************************  Byte ausgeben
                                    IN : A: Byte
                                    OUT: CY=0 für Fehler; A: Fehlernr.
29F8  D5           PUSH   DE
29F9  1E 08        LD     E,08         Zähler für 8 Bits
29FB  57           LD     D,A          Byte
29FC  CB 02        RLC    D            nächstes Bit
29FE  CD 08 2A     CALL   2A08         auf Band schreiben
2A01  30 03        JR     NC,2A06      Fehler ?
2A03  1D           DEC    E            Bitzähler
2A04  20 F6        JR     NZ,29FC      weitere Bits ?
2A06  D1           POP    DE
2A07  C9           RET

**********************************  Bit auf Band schreiben
                                    IN : CY: Bit
                                    OUT: CY=0 für Fehler; A: Fehlernr.
2A08  ED 4B D0 B8  LD     BC,(B8D0)    voriger Zeitwert/Korrekturwert
2A0C  2A D2 B8     LD     HL,(B8D2)    Hauptzeitwert nach L
2A0F  9F           SBC    A            A=$FF, wenn 1-Bit
2A10  67           LD     H,A          für Check-Word retten
2A11  28 07        JR     Z,2A1A       Bit=0 ?
2A13  7D           LD     A,L          sonst Haupt-Zeitwert
2A14  87           ADD    A            *2, da 1-Bit doppelt so lang
2A15  80           ADD    B            Korrektur-Wert addieren
2A16  6F           LD     L,A          als Zeitwert
2A17  79           LD     A,C          Zeitwert des vorigen Bits
2A18  90           SUB    B            Korrekturwert abziehen
2A19  4F           LD     C,A          wieder seten
```

```
2A1A  7D          LD     A,L          neuen Zeitwert
2A1B  32 D0 B8    LD     (B8D0),A     für nächstes Bit speichern
2A1E  2E 0A       LD     L,0A         Port C b5 (WR DATA) gelöscht
2A20  CD 37 2A    CALL   2A37         verzögern und ausgeben
2A23  38 06       JR     C,2A2B       kein Fehler ?
2A25  91          SUB    C            Fehlzeit-urspr. Zeitwert
2A26  30 0C       JR     NC,2A34      Fehlzeit zu groß ?
2A28  2F          CPL                   urspr. Zeitwert-Fehlzeit
2A29  3C          INC    A              (Betrag der Differenz)
2A2A  4F          LD     C,A          als neuen Zeitwert setzen
2A2B  7C          LD     A,H          Flag für 0/1-Bit
2A2C  CD 90 29    CALL   2990         Check-W. entspr. Bit neu setz.
2A2F  2E 0B       LD     L,0B         Port C b5 (WR DATA) gesetzt
2A31  CD 37 2A    CALL   2A37         verzögern und ausgeben
2A34  3E 01       LD     A,01         Fehlernr. f. "Write error a"
2A36  C9          RET

********************************** nächste Halbwelle ausgeben
                                   IN : L=$0A für neg. Flanke
                                        L=$0B für pos. Flanke
                                        C: Zeitwert
                                   OUT: CY=0 für Fehler
                                        A: Fehlzeit
2A37  ED 5F       LD     A,R            Refresh-Zähler durch 2 gibt
2A39  CB 3F       SRL    A              Zeitwert seit letzter Halbw.
2A3B  91          SUB    C            gewünschten Zeitwert abziehen
2A3C  30 03       JR     NC,2A41      Zeit schon abgelaufen ?
2A3E  3C          INC    A              sonst restliche Zeit
2A3F  20 FD       JR     NZ,2A3E        verzögern
2A41  06 F7       LD     B,F7         PIO, Steuerregister
2A43  ED 69       OUT    (C),L        Port C/b5 setzen/löschen
2A45  F5          PUSH   AF           Fehlerflag
2A46  AF          XOR    A              Zeitzähler im Refresh-
2A47  ED 4F       LD     R,A            Register rücksetzen
2A49  F1          POP    AF           Fehlerflags
2A4A  C9          RET

********************************** CAS START MOTOR
                                   OUT: A<b4>: alter Motor-Status
                                        CY=0, Z=1 für Abbruch
2A4B  3E 10       LD     A,10         b4=1 für Motor ein
2A4D  18 02       JR     2A51

********************************** CAS STOP MOTOR
                                   OUT: A<b4>: alter Motor-Status
                                        CY=0, Z=1 für Abbruch
2A4F  3E EF       LD     A,EF         b4=0 für Motor aus

********************************** CAS RESTORE MOTOR
                                   IN : A<b4>:
                                            0 für Motor aus
                                            1 für Motor ein
                                   OUT: A<b4>: alter Motor-Status
                                        CY=0, Z=1 für Abbruch
2A51  C5          PUSH   BC
2A52  06 F6       LD     B,F6         PIO, Port C
2A54  ED 48       IN     C,(C)        alten Motor-Status holen
2A56  04          INC    B            PIO, Steuerregister
```

```
2A57  E6 10      AND   10        eingegebenes Flag
2A59  3E 08      LD    A,08      Code für b4 rücksetzen
2A5B  28 01      JR    Z,2A5E    Flag für Stop ?
2A5D  3C         INC   A         sonst Code für b4 setzen
2A5E  ED 79      OUT   (C),A     Motor-Bit in Port C rücks./s.
2A60  37         SCF             CY=1 für kein Abbruch
2A61  28 0C      JR    Z,2A6F    Motor ausgeschaltet ?
2A63  79         LD    A,C       alter Motor-Status
2A64  E6 10      AND   10        b4 (Motor-Bit) isolieren
2A66  C5         PUSH  BC        alter Motor-Status
2A67  01 C8 00   LD    BC,00C8   Verzögerungswert
2A6A  37         SCF             CY=1 für kein Abbruch
2A6B  CC 72 2A   CALL  Z,2A72    war Motor aus ? dann verzögern
2A6E  C1         POP   BC        alter Motor-Status
2A6F  79         LD    A,C       alten Motor-Status nach A
2A70  C1         POP   BC
2A71  C9         RET

**********************************     Bandhochlaufzeit v., Abbruch prf.
                                       IN : BC: 2. Verzögerungswert
                                       OUT: CY=0, Z=1, A=0 für Abbruch
2A72  C5         PUSH  BC
2A73  E5         PUSH  HL
2A74  CD 89 2A   CALL  2A89        Verzöger. f. Bandhochlaufzeit
2A77  3E 42      LD    A,42        Nr. der Taste ESC
2A79  CD BD 1C   CALL  1CBD        KM TEST KEY, Taste gedrückt ?
2A7C  E1         POP   HL
2A7D  C1         POP   BC
2A7E  20 07      JR    NZ,2A87     dann Flags setzen
2A80  0B         DEC   BC
2A81  78         LD    A,B           sonst verzögern
2A82  B1         OR    C
2A83  20 ED      JR    NZ,2A72
2A85  37         SCF               CY=1 für kein Abbruch
2A86  C9         RET
2A87  AF         XOR   A           CY=0 für Abbruch
2A88  C9         RET

**********************************     Verzögerung für Bandhochlaufzeit
2A89  01 82 06   LD    BC,0682     Verzögerungswert
2A8C  0B         DEC   BC
2A8D  78         LD    A,B           verzögern
2A8E  B1         OR    C
2A8F  20 FB      JR    NZ,2A8C
2A91  C9         RET
2A92  C7         RST   00
2A93  C7         RST   00
2A94  C7         RST   00
2A95  C7         RST   00
2A96  C7         RST   00
2A97  C7         RST   00
```

```
---------------------------- EDITOR ----------------------------

********************************    EDIT
                                    (Eingabezeile holen)
                                    IN : HL: Zeiger auf Eingabebuffer
                                    OUT: HL: Zeiger auf Eingabebuffer
                                         Z=1, wenn ESC
2A98  C5         PUSH  BC
2A99  D5         PUSH  DE
2A9A  E5         PUSH  HL
2A9B  E5         PUSH  HL
2A9C  01 FF 00   LD    BC,00FF        Pos. in Buffer u. Bufferlänge
2A9F  0C         INC   C              Bufferlänge
2AA0  7E         LD    A,(HL)         Zeichen aus Buffer
2AA1  23         INC   HL             Zeiger auf nächstes Zeichen
2AA2  B7         OR    A              Bufferende?
2AA3  20 FA      JR    NZ,2A9F        sonst nächstes Zeichen
2AA5  32 DD B8   LD    (B8DD),A       Insert-Modus ausschalten
2AA8  CD 6F 2C   CALL  2C6F           Copy Cursor ausschalten
2AAB  E1         POP   HL             Zeiger auf Bufferanfang
2AAC  CD 67 2D   CALL  2D67           Buffer ausgeben
2AAF  C5         PUSH  BC             Pos. in Buffer u. Bufferlänge
2AB0  E5         PUSH  HL             Bufferzeiger
2AB1  CD D9 2D   CALL  2DD9           Zeichen von Tastatur holen
2AB4  E1         POP   HL
2AB5  C1         POP   BC
2AB6  CD C6 2A   CALL  2AC6           entspr. Routine anspringen
2AB9  30 F4      JR    NC,2AAF        Zeilenende? dann raus
2ABB  F5         PUSH  AF             Zeichen
2ABC  CD D2 2C   CALL  2CD2           Copy Cursor ausschalten
2ABF  F1         POP   AF             Zeichen
2AC0  E1         POP   HL
2AC1  D1         POP   DE
2AC2  C1         POP   BC
2AC3  FE FC      CP    FC             Editieren mit ESC terminiert?
2AC5  C9         RET

********************************    Editor-Routine anspringen
                                    IN : A: Zeichen
2AC6  E5         PUSH  HL             Bufferzeiger
2AC7  21 E0 2A   LD    HL,2AE0        Tabellenanfang
2ACA  5F         LD    E,A            Zeichen retten
2ACB  78         LD    A,B            Position in Buffer
2ACC  B1         OR    C              Bufferlänge
2ACD  7B         LD    A,E            Zeichen wieder zurück
2ACE  20 0B      JR    NZ,2ADB        Bufferlänge<>0? dann springen
2AD0  FE F0      CP    F0               sonst bei
2AD2  38 07      JR    C,2ADB           Cursortasten
2AD4  FE F4      CP    F4               eine
2AD6  30 03      JR    NC,2ADB          besondere
2AD8  21 1C 2B   LD    HL,2B1C          Tabelle
2ADB  CD F6 2D   CALL  2DF6           durchsuchen
2ADE  E3         EX    (SP),HL        und Routine anspringen
2ADF  C9         RET

********************************    1. Editor-Sprungtabelle
2AE0  13                              Anzahl der Tabelleneinträge
2AE1  01 2C      2C01                 Default: Zeichen in Buffer
```

```
2AE3  FC 42 2B     2B42, CHR$(252)     ESCape
2AE6  EF 40 2B     2B40, CHR$(239)     BRK-Zeichen, ignorieren
2AE9  0D 69 2B     2B69, CHR$(13)      ENTER, Editieren beenden
2AEC  F0 B3 2B     2BB3, CHR$(240)     CURSOR UP
2AEF  F1 7E 2B     2B7E, CHR$(241)     CURSOR DOWN
2AF2  F2 AA 2B     2BAA, CHR$(242)     CURSOR LEFT
2AF5  F3 75 2B     2B75, CHR$(243)     CURSOR RIGHT
2AF8  F8 C7 2B     2BC7, CHR$(248)     CTRL-CURSOR UP
2AFB  F9 92 2B     2B92, CHR$(249)     CTRL-CURSOR DOWN
2AFE  FA BD 2B     2BBD, CHR$(250)     CTRL-CURSOR LEFT
2B01  FB 89 2B     2B89, CHR$(251)     CTRL-CURSOR RIGHT
2B04  F4 A2 2C     2CA2, CHR$(244)     SHIFT-CURSOR UP
2B07  F5 A7 2C     2CA7, CHR$(245)     SHIFT-CURSOR DOWN
2B0A  F6 9D 2C     2C9D, CHR$(246)     SHIFT-CURSOR LEFT
2B0D  F7 98 2C     2C98, CHR$(247)     SHIFT-CURSOR RIGHT
2B10  E0 EA 2C     2CEA, CHR$(224)     COPY
2B13  7F 3D 2C     2C3D, CHR$(127)     DELete
2B16  10 4A 2C     2C4A, CHR$(16)      CLR, Zeich. unt. Cursor lösch.
2B19  E1 F9 2B     2BF9, CHR$(225)     CTRL-TAB, Insert-Flag invert.

*********************************** 2. Editor-Sprungtabelle
                                    (bei leerem Buffer)
2B1C  04                               Anzahl der Tabelleneinträge
2B1D  2B 2B        2B2B                Default: BELL f. Fehler
2B1F  F0 2F 2B     2B2F, CHR$(240)     CURSOR UP
2B22  F1 33 2B     2B33, CHR$(241)     CURSOR DOWN
2B25  F2 3B 2B     2B3B, CHR$(242)     CURSOR LEFT
2B28  F3 37 2B     2B37, CHR$(243)     CURSOR RIGHT

*********************************** BELL
2B2B  3E 07        LD     A,07         Code f. Bell
2B2D  18 0E        JR     2B3D         ausgeben

*********************************** CRSR UP
2B2F  3E 0B        LD     A,0B         Code f. Cursor up
2B31  18 0A        JR     2B3D         ausgeben

*********************************** CRSR DWN
2B33  3E 0A        LD     A,0A         Code f. Line Feed
2B35  18 06        JR     2B3D         ausgeben

*********************************** CRSR RGHT
2B37  3E 09        LD     A,09         Code f. Cursor right
2B39  18 02        JR     2B3D         ausgeben

*********************************** CRSR LEFT
2B3B  3E 08        LD     A,08         Code f. Backspace

2B3D  CD 00 14     CALL   1400         Zeichen ausgeben
2B40  B7           OR     A            CY:=0 f. weiter editieren
2B41  C9           RET

*********************************** ESC
                                    IN : HL: Bufferzeiger
                                    OUT: CY=1 f. Zeilenende
2B42  F5           PUSH   AF           Zeichen retten
2B43  CD 49 2B     CALL   2B49         Buffer u. Abbruchmeldung ausg.
2B46  F1           POP    AF           Zeichen
```

```
2B47  37            SCF                    CY:=1 f. Abbruch
2B48  C9            RET

2B49  CD 69 2B      CALL    2B69           Buffer ausgeben
2B4C  21 61 2B      LD      HL,2B61        Zeiger "*Break*"
2B4F  CD 69 2B      CALL    2B69           ausgeben
2B52  CD 80 11      CALL    1180           Cursorposition holen
2B55  25            DEC     H              Cursor am linken Rand?
2B56  C8            RET     Z              dann zurück
2B57  3E 0D         LD      A,0D             sonst CR
2B59  CD 00 14      CALL    1400             und
2B5C  3E 0A         LD      A,0A             LF
2B5E  C3 00 14      JP      1400             ausgeben

2B61  2A 42 72 65 61 6B 2A 00 *Break*.

*********************************    ENTER
2B69  F5            PUSH    AF              Zeichen retten
2B6A  7E            LD      A,(HL)          Zeichen aus String
2B6B  23            INC     HL              Zeiger nächstes Zeichen
2B6C  B7            OR      A               Bufferende?
?B6D  C4 A8 2D      CALL    NZ,2DA8         sonst Zeichen ausgeben
2B70  20 F8         JR      NZ,2B6A         und nächstes Zeichen
2B72  F1            POP     AF              Zeichen von Tastatur
2B73  37            SCF                     CY:=1 f. Zeilenende
2B74  C9            RET

*********************************    CUR RGHT, Buffer<>0
2B75  16 01         LD      D,01            Zähler f. 1 Zeichen einfügen
2B77  CD 93 2B      CALL    2B93            Zeichen einfügen
2B7A  CA 2B 2B      JP      Z,2B2B          Bufferende? dann BEL f. Fehler
2B7D  C9            RET

*********************************    CUR DOWN, Buffer<>0
2B7E  CD EB 2B      CALL    2BEB            Windowbreite u. Cursorspalte
2B81  79            LD      A,C             Bufferlänge
2B82  90            SUB     B               -Cursorpos=Anz. d. Zeichen
2B83  BA            CP      D               kleiner als Windowbreite?
2B84  DA 2B 2B      JP      C,2B2B          dann BEL f. Fehler
2B87  18 0A         JR      2B93            sonst Zeichen einfügen

*********************************    CTRL-CUR RGHT
2B89  CD EB 2B      CALL    2BEB            Windowbreite u. Cursorspalte
2B8C  7A            LD      A,D             Windowbreite
2B8D  93            SUB     E               -Cursorspalte
2B8E  C8            RET     Z               Cursor am rechten Rand?
2B8F  57            LD      D,A             sonst dorthin
2B90  18 01         JR      2B93            setzen

*********************************    CTRL-CUR DOWN
2B92  51            LD      D,C             Bufferlänge entspr. Zeichen

*********************************    Zeichen in Buffer einfügen
                                     IN : D: Anz d. einzuf. Zeichen
                                     OUT: Z=0, wenn o.k.
2B93  78            LD      A,B             Position in Buffer
2B94  B9            CP      C               Bufferlänge
2B95  C8            RET     Z               Ende erreicht? dann Z=1, raus
```

```
2B96  D5          PUSH  DE
2B97  CD 50 2D    CALL  2D50          Cursor 1 Zeichen nach rechts
2B9A  7E          LD    A,(HL)        aktuelles Zeichen aus Buffer
2B9B  D4 A8 2D    CALL  NC,2DA8       Scrolling? dann Zeichen ausg.
2B9E  04          INC   B             Position in Buffer
2B9F  23          INC   HL            und Bufferzeiger erhöhen
2BA0  D4 67 2D    CALL  NC,2D67       o.k.? dann Rest d. Buff. ausg.
2BA3  D1          POP   DE
2BA4  15          DEC   D             Zähler f. Zeichen
2BA5  20 EC       JR    NZ,2B93       <>0? dann nächstes Zeichen
2BA7  F6 FF       OR    FF            Z:=0 f. o.k.
2BA9  C9          RET

*********************************** CUR LEFT, Buffer<>0
2BAA  16 01       LD    D,01            1 Zeichen
2BAC  CD C8 2B    CALL  2BC8            zurück
2BAF  CA 2B 2B    JP    Z,2B2B          Fehler? dann BEL
2BB2  C9          RET

*********************************** CUR UP, Buffer<>0
2BB3  CD EB 2B    CALL  2BEB            Windowbreite u. Cursorspalte
2BB6  78          LD    A,B             Position in Buffer
2BB7  BA          CP    D               < Windowbreite?
2BB8  DA 2B 2B    JP    C,2B2B          dann BEL f. Fehler
2BBB  18 0B       JR    2BC8            an Bufferanfang zurückgehen

*********************************** CTRL-CUR LEFT
2BBD  CD EB 2B    CALL  2BEB            Windowbreite u. Cursorspalte
2BC0  7B          LD    A,E             Cursorspalte
2BC1  D6 01       SUB   01              Cursor am linken Rand?
2BC3  C8          RET   Z               dann zurück
2BC4  57          LD    D,A             sonst Cursor an linken Rand
2BC5  18 01       JR    2BC8            setzen

*********************************** CTRL-CUR UP
2BC7  51          LD    D,C             Cursor an Bufferanfang

*********************************** Cursor in Buffer zurück
                                      IN : D: Zeichenzahl
                                      OUT: Z=0 f. Fehler
2BC8  78          LD    A,B             Cursor
2BC9  B7          OR    A               am Bufferanfang?
2BCA  C8          RET   Z               dann zurück, Z:=1
2BCB  CD 4A 2D    CALL  2D4A            Cursor 1 Zeichen nach links
2BCE  30 07       JR    NC,2BD7         über linken Rand? d. Copy Cur.
2BD0  05          DEC   B               Pos. in Buffer
2BD1  2B          DEC   HL              Bufferzeiger
2BD2  15          DEC   D               Zeichenzahl
2BD3  20 F3       JR    NZ,2BC8         noch Zeichen? dann bearbeiten
2BD5  18 11       JR    2BE8            sonst Z:=0, raus
2BD7  78          LD    A,B             Pos. in Buffer
2BD8  B7          OR    A               Bufferanfang erreicht?
2BD9  28 0A       JR    Z,2BE5          dann Buffer ausgeben, raus
2BDB  05          DEC   B               Position in Buffer
2BDC  2B          DEC   HL              Bufferzeiger
2BDD  D5          PUSH  DE              Zeichenzahl retten
2BDE  CD 29 2D    CALL  2D29            Copy Cursor nach rechts
2BE1  D1          POP   DE              Zeichenzahl
```

```
2BE2  15          DEC   D           herunterzählen
2BE3  20 F2       JR    NZ,2BD7     noch Zeichen? dann bearbeiten
2BE5  CD 67 2D    CALL  2D67        Buffer ausgeben
2BE8  F6 FF       OR    FF          Z:=0 f. o.k.
2BEA  C9          RET

*********************************  Windowbr. u. Cursorsp.
                                   OUT: D: Windowbreite
                                        E: Cursorspalte
2BEB  E5          PUSH  HL          Bufferzeiger retten
2BEC  CD 56 12    CALL  1256        Windowgrenzen holen
2BEF  7A          LD    A,D         rechte Grenze
2BF0  94          SUB   H           - linke Grenze
2BF1  3C          INC   A           + 1
2BF2  57          LD    D,A         = Breite des Windows
2BF3  CD 80 11    CALL  1180        Cursorposition
2BF6  5C          LD    E,H         Spaltenposition nach E
2BF7  E1          POP   HL          Bufferzeiger wiederholen
2BF8  C9          RET

*********************************  CTRL-TAB
2BF9  3A DD B8    LD    A,(B8DD)    Insert-Flag
2BFC  2F          CPL               komplementieren
2BFD  32 DD B8    LD    (B8DD),A    und neu setzen
2C00  C9          RET

*********************************  Zeichen in Buffer schreiben
                                   IN : A: Zeichen
2C01  B7          OR    A           Zeichen
2C02  C8          RET   Z           Null? dann nichts tun, zurück
2C03  5F          LD    E,A         Zeichen retten
2C04  3A DD B8    LD    A,(B8DD)    Insert-Flag
2C07  B7          OR    A           Insert-Modus an?
2C08  28 0D       JR    Z,2C17      dann Zeichen einfügen
2C0A  78          LD    A,B         sonst Cursor
2C0B  B9          CP    C           am Bufferende?
2C0C  28 09       JR    Z,2C17      dann Zeichen einfügen
2C0E  73          LD    (HL),E      sonst Zeichen einfach in Buff.
2C0F  7B          LD    A,E         Zeichen zurück
2C10  CD A8 2D    CALL  2DA8        und ausgeben
2C13  23          INC   HL          Bufferzeiger
2C14  04          INC   B           und Pos. in Buffer erhöhen
2C15  B7          OR    A           CY:=0 f. o.k.
2C16  C9          RET

*********************************  Zeichen in Buffer einfügen
                                   IN : E: Zeichen
2C17  79          LD    A,C         Bufferlänge
2C18  FE FF       CP    FF          bereits maximal?
2C1A  CA 2B 2B    JP    Z,2B2B      dann BEL f. Fehler
2C1D  AF          XOR   A           Flag f. Copy Cursor=Cursor
2C1E  32 DC B8    LD    (B8DC),A    setzen
2C21  7B          LD    A,E         Zeichen
2C22  CD A8 2D    CALL  2DA8        ausgeben
2C25  0C          INC   C           Bufferlänge erhöhen
2C26  E5          PUSH  HL          Bufferzeiger retten
2C27  7E          LD    A,(HL)      Zeichen aus Buffer
2C28  73          LD    (HL),E      neues Zeichen in Buffer
```

```
2C29  5F          LD     E,A          Zeichen als neues Zeich. setz.
2C2A  23          INC    HL           Bufferzeig. auf nächst. Zeich.
2C2B  B7          OR     A            Bufferende?
2C2C  20 F9       JR     NZ,2C27      sonst nächstes Zeichen
2C2E  77          LD     (HL),A       Null an Bufferende
2C2F  E1          POP    HL           alter Bufferzeiger
2C30  04          INC    B            Pos. in Buffer erhöhen
2C31  23          INC    HL           Zeiger auf nächstes Zeichen
2C32  CD 67 2D    CALL   2D67         restl. Buffer ausgeben
2C35  3A DC B8    LD     A,(B8DC)     Cursor=Copy Cursor
2C38  B7          OR     A            testen
2C39  C4 29 2D    CALL   NZ,2D29      wenn ungl., d. Copy Cur. rech.
2C3C  C9          RET

*********************************  DEL
2C3D  78          LD     A,B          Position in Buffer
2C3E  B7          OR     A            am Bufferanfang?
2C3F  CA 2B 2B    JP     Z,2B2B       dann BEL f. Fehler
2C42  CD 4A 2D    CALL   2D4A         1 Zeichen nach links
2C45  D2 2B 2B    JP     NC,2B2B      über linken Rand? dann BEL f. Fehler
2C48  05          DEC    B            Position in Buffer
2C49  2B          DEC    HL           und Bufferzeiger erniedrigen

*********************************  CLR
2C4A  78          LD     A,B          Position in Buffer
2C4B  B9          CP     C            am Bufferende?
2C4C  CA 2B 2B    JP     Z,2B2B       dann BEL f. Fehler
2C4F  E5          PUSH   HL           Bufferzeiger retten
2C50  23          INC    HL           Zeiger nächstes Zeichen
2C51  7E          LD     A,(HL)       Zeichen laden
2C52  2B          DEC    HL           Zeiger auf letztes Zeichen
2C53  77          LD     (HL),A       Zeichen eine Stelle vorrücken
2C54  23          INC    HL           Zeiger nächstes Zeichen
2C55  B7          OR     A            Zeilenende erreicht?
2C56  20 F8       JR     NZ,2C50      sonst nächstes Zeichen
2C58  2B          DEC    HL           Zeiger auf letztes Zeichen
2C59  36 20       LD     (HL),20      SPC als letztes Zeichen
2C5B  32 DC B8    LD     (B8DC),A     Copy Cursor:=Cursor
2C5E  E3          EX     (SP),HL      alten Bufferzeiger
2C5F  CD 67 2D    CALL   2D67         restl. Buffer ausgeben
2C62  E3          EX     (SP),HL      Zeiger auf letztes Zeichen
2C63  36 00       LD     (HL),00      Null f. Bufferende setzen
2C65  E1          POP    HL           alter Bufferzeiger
2C66  0D          DEC    C            Bufferlänge
2C67  3A DC B8    LD     A,(B8DC)     Flag f. Cursorgleichheit
2C6A  B7          OR     A            Cursor ungleich?
2C6B  C4 2D 2D    CALL   NZ,2D2D      dann Copy Cursor nach links
2C6E  C9          RET

*********************************  Copy Cursor ausschalten
2C6F  21 00 00    LD     HL,0000      $0000
2C72  22 DE B8    LD     (B8DE),HL    als CC-Koordinaten setzen
2C75  C9          RET
```

```
**********************************    beide Cursor vergleichen
                                      IN : HL: Koordinaten Edit Cur.
                                      OUT: Z=CY=1, wenn gleich
                                           DE: Koordinaten Copy Cur.
2C76  ED 5B DE B8  LD     DE,(B8DE)     CC-Koordinaten
2C7A  7C           LD     A,H           EC-Spalte
2C7B  AA           XOR    D             =CC-Spalte?
2C7C  C0           RET    NZ            sonst Z=CY=0, zurück
2C7D  7D           LD     A,L           EC-Zeile
2C7E  AB           XOR    E             =CC-Zeile?
2C7F  C0           RET    NZ            sonst Z=CY=0, zurück
2C80  37           SCF                  Z=CY=1 bei Gleichheit
2C81  C9           RET

**********************************    neue CC-Zeile berechnen
                                      IN : A:Scrolling Differenz
2C82  4F           LD     C,A           gescrollte Zeilen
2C83  2A DE B8     LD     HL,(B8DE)     CC-Koordinaten
2C86  7C           LD     A,H             Copy Cursor
2C87  B5           OR     L               ausgeschaltet?
2C88  C8           RET    Z             dann zurück
2C89  7D           LD     A,L           CC-Zeilenposition
2C8A  81           ADD    C             Scrolling Differenz addieren
2C8B  6F           LD     L,A           ergibt neue Zeile
2C8C  CD CE 11     CALL   11CE          Cursor in erlaubte Grenzen
2C8F  38 03        JR     C,2C94        o.k.? dann wieder abspeichern
2C91  21 00 00     LD     HL,0000       sonst ausschalten
2C94  22 DE B8     LD     (B8DE),HL     neue Koordinaten setzen
2C97  C9           RET

**********************************    SHIFT-CUR RGHT
2C98  11 00 01     LD     DE,0100       Offset f. Copy Cursor
2C9B  18 0D        JR     2CAA          Copy Cursor bewegen

**********************************    SHIFT-CUR LEFT
2C9D  11 00 FF     LD     DE,FF00       Offset f. Copy Cursor
2CA0  18 08        JR     2CAA          Copy Cursor bewegen

**********************************    SHIFT-CUR UP
2CA2  11 FF 00     LD     DE,00FF       Offset f. Copy Cursor
2CA5  18 03        JR     2CAA          Copy Cursor bewegen

**********************************    SHIFT-CUR DOWN
2CA7  11 01 00     LD     DE,0001       Offset f. Copy Cursor

**********************************    Copy Cursor bewegen
                                      IN : D: Spaltenoffset
                                           E: Zeilenoffset
2CAA  C5           PUSH   BC
2CAB  E5           PUSH   HL
2CAC  2A DE B8     LD     HL,(B8DE)     CC-Koordinaten
2CAF  7C           LD     A,H             Copy Cursor
2CB0  B5           OR     L               ausgeschaltet?
2CB1  CC 80 11     CALL   Z,1180        dann CC auf Edit Cursor setzen
2CB4  7C           LD     A,H           CC-Spalte
2CB5  82           ADD    D             Spaltenoffset addieren
2CB6  67           LD     H,A
2CB7  7D           LD     A,L           CC-Zeile
```

```
2CB8  83          ADD   E            Zeilenoffset addieren
2CB9  6F          LD    L,A
2CBA  CD CE 11    CALL  11CE         und in Windowgrenzen bringen
2CBD  30 0B       JR    NC,2CCA      Scrolling? dann raus, alte Pos.
2CBF  E5          PUSH  HL           Koordinaten retten
2CC0  CD D2 2C    CALL  2CD2         CC an lfd. Pos. ausschalten
2CC3  E1          POP   HL           neue Koordinaten
2CC4  22 DE B8    LD    (B8DE),HL    setzen
2CC7  CD CD 2C    CALL  2CCD         und CC wieder anschalten
2CCA  E1          POP   HL
2CCB  C1          POP   BC
2CCC  C9          RET

**********************************  Copy Cursor anschalten
2CCD  11 68 12    LD    DE,1268        Adr. f. Cursor anschalten
2CD0  18 03       JR    2CD5           Cursor anschalten

**********************************  Copy Cursor ausschalten
2CD2  11 68 12    LD    DE,1268        Adr. f. Cursor ausschalten

**********************************  Copy Cursor umschalten
                                    IN : DE: Adr. d. Umschaltroutine
2CD5  2A DE B8    LD    HL,(B8DE)      CC-Koordinaten
2CD8  7C          LD    A,H              Copy Cursor
2CD9  B5          OR    L                inaktiv?
2CDA  C8          RET   Z              dann zurück
2CDB  E5          PUSH  HL             Koordinaten retten
2CDC  CD 80 11    CALL  1180           EC-Koordinaten holen
2CDF  E3          EX    (SP),HL        CC-Koordinaten nach HL
2CE0  CD 74 11    CALL  1174           und setzen
2CE3  CD 16 00    CALL  0016           Copy Cursor umschalten
2CE6  E1          POP   HL             EC-Koordinaten
2CE7  C3 74 11    JP    1174           setzen

**********************************  COPY
2CEA  C5          PUSH  BC
2CEB  E5          PUSH  HL
2CEC  CD 80 11    CALL  1180           aktuelle EC-Koordinaten holen
2CEF  EB          EX    DE,HL          und nach DE
2CF0  2A DE B8    LD    HL,(B8DE)      CC-Koordinaten
2CF3  7C          LD    A,H            Copy Cursor
2CF4  B5          OR    L              aktiv?
2CF5  20 0C       JR    NZ,2D03        dann von dort kopieren
2CF7  78          LD    A,B            sonst Buffer
2CF8  B1          OR    C              nicht leer?
2CF9  20 26       JR    NZ,2D21        dann raus
2CFB  CD 80 11    CALL  1180           aktuelle EC-Position
2CFE  22 DE B8    LD    (B8DE),HL      als CC-Koordinaten setzen
2D01  18 06       JR    2D09           da EC schon an, nicht umschalten
2D03  CD 74 11    CALL  1174           Cursor neu setzen
2D06  CD 68 12    CALL  1268           Cursor anschalten
2D09  CD AB 13    CALL  13AB           Zeichen unter Cursor holen
2D0C  F5          PUSH  AF             und retten
2D0D  EB          EX    DE,HL          EC-Position nach HL
2D0E  CD 74 11    CALL  1174           und wieder setzen
2D11  2A DE B8    LD    HL,(B8DE)      CC-Position holen
2D14  24          INC   H              CC-Spalte erhöhen
2D15  CD CE 11    CALL  11CE           und in Window zwingen
```

```
2D18  30 03     JR    NC,2D1D      über Rand hinaus?
2D1A  22 DE B8  LD    (B8DE),HL    sonst neue CC-Pos. setzen
2D1D  CD CD 2C  CALL  2CCD         Copy cursor anschalten
2D20  F1        POP   AF           Zeichen
2D21  E1        POP   HL
2D22  C1        POP   BC
2D23  DA 01 2C  JP    C,2C01       Zeich. identif.? d. in Buffer
2D26  C3 2B 2B  JP    2B2B         sonst BEL f. Fehler

*********************************  Copy Cursor nach rechts
2D29  16 01     LD    D,01           CC-Spaltenoffset
2D2B  18 02     JR    2D2F           Copy Cursor verschieben

*********************************  Copy Cursor nach links
2D2D  16 FF     LD    D,FF           CC-Spaltenoffset

*********************************  Copy Cursor verschieben
                                   IN : D: Spaltenofffset
2D2F  C5        PUSH  BC
2D30  E5        PUSH  HL
2D31  D5        PUSH  DE             Offset retten
2D32  CD D2 2C  CALL  2CD2           Copy Cursor ausschalten
2D35  D1        POP   DE             Offset
2D36  2A DE B8  LD    HL,(B8DE)      CC-Koordinaten
2D39  7C        LD    A,H              Copy Cursor
2D3A  B5        OR    L                inaktiv?
2D3B  28 09     JR    Z,2D46         dann raus
2D3D  7C        LD    A,H            sonst CC-Spalte
2D3E  82        ADD   D              und Offset addieren
2D3F  67        LD    H,A            als neue Spalte
2D40  CD 8C 2C  CALL  2C8C           in Window zwingen u. speich.
2D43  CD CD 2C  CALL  2CCD           Copy Cursor wieder anschalten
2D46  E1        POP   HL
2D47  C1        POP   BC
2D48  B7        OR    A              CY:=0
2D49  C9        RET

*********************************  Cursor nach links
2D4A  D5        PUSH  DE
2D4B  11 08 FF  LD    DE,FF08        Spaltenoffset und BS
2D4E  18 04     JR    2D54           Cursor verschieben

*********************************  Cursor nach rechts
2D50  D5        PUSH  DE
2D51  11 09 01  LD    DE,0109        Spaltenoffset und TAB

*********************************  Cursor verschieben
                                   IN : D: Spaltenoffset; E: Zeichen (08,09)
                                   OUT: CY=0, wenn o.k.
2D54  C5        PUSH  BC
2D55  E5        PUSH  HL
2D56  CD 80 11  CALL  1180           aktuelle Cursorposition
2D59  7A        LD    A,D            Offset
2D5A  84        ADD   H              und Cursorspalte addieren
2D5B  67        LD    H,A            als neue Spalte setzen
2D5C  CD CE 11  CALL  11CE           und in Fenstergrenzen zwingen
2D5F  7B        LD    A,E            Zeichen
2D60  DC 00 14  CALL  C,1400         Spalte o.k.? d. Zeich. ausg.
```

```
2D63  E1          POP   HL
2D64  C1          POP   BC
2D65  D1          POP   DE
2D66  C9          RET

********************************** Buffer ab HL ausgeben
2D67  C5          PUSH  BC
2D68  E5          PUSH  HL        Bufferzeiger retten
2D69  EB          EX    DE,HL     und nach DE
2D6A  CD 80 11    CALL  1180      Cursorposition holen
2D6D  4F          LD    C,A       Scrolling Zähler
2D6E  EB          EX    DE,HL
2D6F  7E          LD    A,(HL)    Zeichen aus Buffer
2D70  23          INC   HL        Zeiger auf nächstes Zeichen
2D71  B7          OR    A         Bufferende?
2D72  C4 85 2D    CALL  NZ,2D85   sonst Zeichen ausgeben
2D75  20 F8       JR    NZ,2D6F   und nächstes Zeichen holen
2D77  CD 80 11    CALL  1180      Cursorposition holen
2D7A  91          SUB   C         Scrolling Differenz berechnen
2D7B  EB          EX    DE,HL     alte Cursorposition nach HL
2D7C  85          ADD   L         Scrol. Dif. + alte Cursorzeile
2D7D  6F          LD    L,A       als neue Zeile setzen
2D7E  CD 74 11    CALL  1174      neue Position setzen
2D81  E1          POP   HL
2D82  C1          POP   BC
2D83  B7          OR    A         CY:=0 f. o.k.
2D84  C9          RET

********************************** Zeichen ausgeben
                                  IN : A: Zeichen
                                       C: Scrolling Zähler
                                       DE: Cursorpos.
2D85  F5          PUSH  AF        Zeichen
2D86  C5          PUSH  BC
2D87  D5          PUSH  DE
2D88  E5          PUSH  HL
2D89  47          LD    B,A       Zeichen retten
2D8A  CD 80 11    CALL  1180      Cursorpos. holen
2D8D  91          SUB   C         Scrolling Differ. berechnen
2D8E  83          ADD   E         + alte Cursorzeile
2D8F  5F          LD    E,A       = neue Cursorzeile
2D90  48          LD    C,B       Zeichen
2D91  CD CE 11    CALL  11CE      Cursor in Fenstergrenzen
2D94  38 05       JR    C,2D9B    o.k.? dann Cur. in DE in Win.
2D96  78          LD    A,B       sonst Flag f. hoch-scrollen
2D97  87          ADD   A           A:=FF=-1, wenn nach oben
2D98  3C          INC   A           A:=01= 1, wenn nach unten
2D99  83          ADD   E         zu Cursorzeile addieren
2D9A  5F          LD    E,A       ergibt neue Cursorzeile
2D9B  EB          EX    DE,HL     nach HL
2D9C  CD CE 11    CALL  11CE      in gültige Grenzen bringen
2D9F  79          LD    A,C       Zeichen
2DA0  DC A8 2D    CALL  C,2DA8    o.k.? dann ausg., ggf. scrol.
2DA3  E1          POP   HL
2DA4  D1          POP   DE
2DA5  C1          POP   BC
2DA6  F1          POP   AF
2DA7  C9          RET
```

```
**********************************   Zeichen ausgeben
                                     IN : A: Zeichen
2DA8  F5          PUSH   AF
2DA9  C5          PUSH   BC
2DAA  D5          PUSH   DE
2DAB  E5          PUSH   HL
2DAC  47          LD     B,A           Zeichen retten
2DAD  CD 80 11    CALL   1180          Cursorposition holen
2DB0  4F          LD     C,A           Scrolling Zähler retten
2DB1  C5          PUSH   BC            mit Zeichen auf Stack
2DB2  CD CE 11    CALL   11CE          Cur. in gült. Grenzen bringen
2DB5  C1          POP    BC            Zeichen und Scrolling Zähler
2DB6  DC 76 2C    CALL   C,2C76        über Rand? sonst Curs. vergl.
2DB9  F5          PUSH   AF            Vergleichsergebnis retten
2DBA  DC D2 2C    CALL   C,2CD2        wenn gleich, Copy Cursor an
2DBD  78          LD     A,B           Zeichen
2DBE  C5          PUSH   BC            und Zähler retten
2DBF  CD 34 13    CALL   1334          Zeichen ausg., Cursor weiter
2DC2  C1          POP    BC            Zähler u. Zeichen
2DC3  CD 80 11    CALL   1180          Cursorposition holen
2DC6  91          SUB    C             Scrolling Differenz berechnen
2DC7  C4 82 2C    CALL   NZ,2C82       Scrolling? dann neue Zeile
2DCA  F1          POP    AF            Flag f. Cursorvergleich
2DCB  30 07       JR     NC,2DD4       waren ungleich? dann raus
2DCD  9F          SBC    A             $FF als Flag f. Ungleichheit
2DCE  32 DC B8    LD     (B8DC),A      setzen
2DD1  CD CD 2C    CALL   2CCD          Copy Cursor anschalten
2DD4  E1          POP    HL
2DD5  D1          POP    DE
2DD6  C1          POP    BC
2DD7  F1          POP    AF
2DD8  C9          RET

**********************************   Zeichen von Tastatur holen
                                     OUT: A: Zeichen
2DD9  CD 80 11    CALL   1180          Cursorposition holen
2DDC  4F          LD     C,A           Scrolling Zähler
2DDD  CD CE 11    CALL   11CE          Cursor in gültige Grenzen
2DE0  CD 76 2C    CALL   2C76          EC und CC vergleichen
2DE3  DA 3C 1A    JP     C,1A3C        gleich? dann Zeichen holen
2DE6  CD 79 12    CALL   1279          Edit Cursor anschalten
2DE9  CD 80 11    CALL   1180          neue Cursorposition holen
2DEC  91          SUB    C             Scrolling Dif. berechnen
2DED  C4 82 2C    CALL   NZ,2C82       ggf. neue Zeile berechnen
2DF0  CD 3C 1A    CALL   1A3C          Zeichen holen
2DF3  C3 81 12    JP     1281          Cursor ausschalten

**********************************   Adr. aus Tabelle holen
                                     IN : A: Zeichen
                                          HL: Tabellenanfang
                                     OUT: HL: entspr. Adresse
2DF6  F5          PUSH   AF
2DF7  C5          PUSH   BC
2DF8  46          LD     B,(HL)        Anz. d. Tabelleneinträge
2DF9  23          INC    HL            Zeiger auf Default-Adresse
2DFA  E5          PUSH   HL            retten
2DFB  23          INC    HL              Default-Adresse
2DFC  23          INC    HL              übergehen
```

```
2DFD  BE        CP    (HL)       Tab.-Byte m. ges. Byte vergl.
2DFE  23        INC   HL         Zeiger auf entspr. Adresse
2DFF  28 04     JR    Z,2E05     Bytes gleich? dann gefunden
2E01  05        DEC   B          sonst Zähler f. Einträge
2E02  20 F7     JR    NZ,2DFB    noch nicht null? dann weiter
2E04  E3        EX    (SP),HL    sonst Zeiger auf Def. nach HL
2E05  F1        POP   AF         Zeiger vom Stack löschen
2E06  7E        LD    A,(HL)       entspr.
2E07  23        INC   HL           Adresse
2E08  66        LD    H,(HL)       nach
2E09  6F        LD    L,A          HL laden
2E0A  C1        POP   BC
2E0B  F1        POP   AF
2E0C  C9        RET

2E0D  C7        RST   00
2E0E  C7        RST   00
2E0F  C7        RST   00
2E10  C7        RST   00
2E11  C7        RST   00
2E12  C7        RST   00
2E13  C7        RST   00
2E14  C7        RST   00
2E15  C7        RST   00
2E16  C7        RST   00
2E17  C7        RST   00
```

```
----------------- FLOATING POINT ARITHMETICS (FLO) -----------------

**********************************   FLO Zahl kopieren
                                     IN : HL: Zieladresse
                                          DE: Quelladresse
                                     OUT: A: Exponent
2E18  E5         PUSH  HL
2E19  D5         PUSH  DE
2E1A  C5         PUSH  BC
2E1B  EB         EX    DE,HL
2E1C  01 05 00   LD    BC,0005         Anz. d. zu kopierenden Bytes
2E1F  ED B0      LDIR                  Zahl kopieren
2E21  EB         EX    DE,HL
2E22  2B         DEC   HL              Zeiger auf Exponenten
2E23  7E         LD    A,(HL)          Exponenten laden
2E24  C1         POP   BC
2E25  D1         POP   DE
2E26  E1         POP   HL
2E27  37         SCF
2E28  C9         RET

**********************************   FLO INT>REAL
                                     IN : HL: Integer, unsigned
                                          A: Vorzeichen
                                          (DE): Zieladr. f. FLO-Zahl
                                     OUT: (DE): normal. FLO-Zahl
2E29  D5         PUSH  DE
2E2A  C5         PUSH  BC
2E2B  F6 7F      OR    7F              alle Bits außer Vorz. setzen
2E2D  47         LD    B,A             und retten
2E2E  AF         XOR   A
2E2F  12         LD    (DE),A            Lo-Bytes
2E30  13         INC   DE                des FLO-Akkus
2E31  12         LD    (DE),A            löschen
2E32  13         INC   DE
2E33  0E 90      LD    C,90            Exp. f. b0 d. 2.MSB =1
2E35  7C         LD    A,H             Hi-Byte d. Integer
2E36  B7         OR    A               null?
2E37  20 08      JR    NZ,2E41
2E39  4F         LD    C,A             Exponent löschen
2E3A  65         LD    H,L             Lo-Byte ins Hi-Byte schieben
2E3B  6F         LD    L,A             Lo-Byte löschen
2E3C  B4         OR    H               Lo-Byte null?
2E3D  28 0D      JR    Z,2E4C          dann FLO-Zahl null, raus
2E3F  0E 88      LD    C,88            Exp. f. b0 d. 1.MSB =1
2E41  FA 4B 2E   JP    M,2E4B          b7=1? dann raus
2E44  29         ADD   HL,HL           Zahl nach oben schieben
2E45  0D         DEC   C               Exp. zum Ausgleich erniedrigen
2E46  B4         OR    H
2E47  F2 44 2E   JP    P,2E44          b7=0? d. weiter normalisieren
2E4A  7C         LD    A,H               in das MSB
2E4B  A0         AND   B                 das Vorzeichen der Zahl
2E4C  EB         EX    DE,HL
2E4D  73         LD    (HL),E          2.MSB abspeichern
2E4E  23         INC   HL              Zeiger auf 1.MSB
2E4F  77         LD    (HL),A          1.MSB (m. Vorz.) abspeichern
2E50  23         INC   HL              Zeiger auf Exponent
2E51  71         LD    (HL),C          Exponenten abspeichern
```

```
2E52  C1          POP    BC
2E53  E1          POP    HL
2E54  C9          RET

**********************************  FLO 4-Bytes>REAL
                                    IN : (HL): 4-Byte Integer
                                    OUT: (HL): entspr. FLO-Zahl
2E55  C5          PUSH   BC
2E56  01 00 A0    LD     BC,A000      Exp. f. 32 Bit, Rundungsbyte=0
2E59  CD 60 2E    CALL   2E60         Zahl wandeln
2E5C  C1          POP    BC
2E5D  C9          RET

**********************************  FLO 5-Bytes>REAL
                                    IN : (HL): 4 MSB d. Integer
                                            C: LSB d. Integer
                                    OUT: (HL): entspr. FLO-Zahl
2E5E  06 A8       LD     B,A8         Exp. f. 40 Bit
2E60  D5          PUSH   DE
2E61  CD A1 36    CALL   36A1         5-Byte wandeln nach Real
2E64  D1          POP    DE
2E65  C9          RET

**********************************  FLO REAL>INTEGER
                                    IN : (HL): FLO-Zahl
                                    OUT: HL: Integer
                                         CY=0, wenn Überlauf
                                         Z=1, A=0, wenn HL=0
                                         A: Vorzeichen
2E66  E5          PUSH   HL           Zeiger auf Real-Zahl
2E67  DD E1       POP    IX           nach IX
2E69  AF          XOR    A
2E6A  DD 96 04    SUB    (IX+04)      Exp. ins Zweierkomplement
2E6D  28 1B       JR     Z,2E8A       =0? dann Zahl=0, raus
2E6F  C6 90       ADD    90           =2er Kompl. d. FAC-Bitstellen
2E71  D0          RET    NC           Exp war >$90? dann Überlauf
2E72  D5          PUSH   DE
2E73  C5          PUSH   BC
2E74  C6 10       ADD    10           16-A=Anz. d. zu shift. Bits
2E76  CD 3D 36    CALL   363D         FAC rechtsverschieben
2E79  CB 21       SLA    C            Rundungsbit
2E7B  ED 5A       ADC    HL,DE        auf Ergebnis addieren
2E7D  28 08       JR     Z,2E87       Erg.=0 o. Überl.? d. behand.
2E7F  DD 7E 03    LD     A,(IX+03)    höchstes Mantissenbyte
2E82  B7          OR     A            Flags nach Vorzeichen setzen
2E83  3F          CCF                 CY:=1 f. o.k.
2E84  C1          POP    BC
2E85  D1          POP    DE
2E86  C9          RET
2E87  9F          SBC    A            CY beibeh., A:=$FF f. Fehler
2E88  18 F9       JR     2E83         raus
2E8A  6F          LD     L,A            Integer
2E8B  67          LD     H,A            gleich null
2E8C  37          SCF                 CY:=1 f. o.k.
2E8D  C9          RET
```

```
**********************************   FLO Zahl runden
                                     IN : (HL): FLO-Zahl
                                     OUT: (HL): ger. 4-Byte Integer
                                          CY=1, wenn o.k.
                                          C: Länge signif. Mantisse
2E8E  CD A1 2E     CALL   2EA1          Nachkommastellen abschneiden
2E91  D0           RET    NC            Fehler? d. CY=0 f. Fehl., raus
2E92  F0           RET    P             Rundungsbit =0? d. o.k., raus
2E93  E5           PUSH   HL            sonst Zeiger auf Zahl retten
2E94  79           LD     A,C           Anz. signifik. Mantissenbytes
2E95  34           INC    (HL)          Mantissenbyte aufrunden
2E96  20 06        JR     NZ,2E9E       kein Übertrag? dann o.k.
2E98  23           INC    HL            sonst Zeiger auf nächstes Byte
2E99  3D           DEC    A             Anz. d. Mantissenbytes
2E9A  20 F9        JR     NZ,2E95       noch nicht null? d. weiter
2E9C  34           INC    (HL)          höchstes signif. Byte aufr.
2E9D  0C           INC    C             Anz. signif. Mantissenbytes
2E9E  E1           POP    HL            Zeiger auf Zahl
2E9F  37           SCF                  CY:=1 f. o.k.
2EA0  C9           RET

**********************************   FLO Nachkommastellen abschneiden
                                     IN : (HL): FLO-Zahl
                                     OUT: (HL): 4-Byte Integer
                                          B: isoliertes Vorzeichen
                                          C: Anz. signif. Mantissenb.
                                          A: Rundungsbyte
                                          S: Rundungsbit
2EA1  E5           PUSH   HL            Zeiger auf Zahl
2EA2  D5           PUSH   DE
2EA3  E5           PUSH   HL            Zeiger auf Zahl
2EA4  DD E1        POP    IX            nach IX
2EA6  CD 04 36     CALL   3604          Nachkommastellen abschneiden
2EA9  D1           POP    DE
2EAA  E1           POP    HL
2EAB  C9           RET

**********************************   FLO INT-Funktion
                                     IN : (HL): FLO-Zahl
                                     OUT: (HL): 4-Byte Integer
2EAC  CD A1 2E     CALL   2EA1          Nachkommastellen abschneiden
2EAF  D0           RET    NC            Fehler? dann raus
2EB0  C8           RET    Z             Zahl =0? dann raus
2EB1  CB 78        BIT    7,B           Zahl positiv?
2EB3  C8           RET    Z             dann raus
2EB4  18 DD        JR     2E93          sonst 1 addieren

**********************************   FLO Zahl f. Dez.-Wandlung aufber.
                                     IN : (HL): FLO-Zahl
                                     OUT: HL: Zeiger 1.MSB d. Mantisse
                                          E: Kommaposition
                                          C: Anz. sign. Mantissenbytes
                                          B: Vorzeichen
2EB6  CD E8 35     CALL   35E8          Vorzeichen d. Zahl
2EB9  47           LD     B,A           nach B
2EBA  28 52        JR     Z,2F0E        Zahl=0? dann null setzen, raus
2EBC  FC FB 35     CALL   M,35FB        Zahl neg.? d. Vorz. invert.
2EBF  E5           PUSH   HL
```

```
2EC0  DD 7E 04     LD     A,(IX+04)    Exponent
2EC3  D6 80        SUB    80           realer Exp. (im 2er Kompl.)
2EC5  5F           LD     E,A          nach E
2EC6  9F           SBC    A            A:=FF, wenn Zahl<1, sonst A:=0
2EC7  57           LD     D,A          Flag nach D
2EC8  6B           LD     L,E          realer Exponent
2EC9  62           LD     H,D          Flag f. Exp. negativ
2ECA  29           ADD    HL,HL
2ECB  29           ADD    HL,HL          den
2ECC  29           ADD    HL,HL          vorzeichenerweiterten
2ECD  19           ADD    HL,DE          16-Bit Exponenten
2ECE  29           ADD    HL,HL          mit
2ECF  19           ADD    HL,DE          77 (ca. 256*lg2)
2ED0  29           ADD    HL,HL          multiplizieren,
2ED1  29           ADD    HL,HL          d.h. Binärexponenten
2ED2  19           ADD    HL,DE          in Dezimalexponenten wandeln
2ED3  7C           LD     A,H          Hi-Byte, also Exp*77/256, *lg2
2ED4  D6 09        SUB    09           =Zehnerstellen-9
2ED6  5F           LD     E,A          nach E
2ED7  E1           POP    HL           Zeiger auf Zahl
2ED8  C5           PUSH   BC
2ED9  D5           PUSH   DE
2EDA  C4 1F 2F     CALL   NZ,2F1F      wenn dez. Exp. <>9, Zahl/10^A
2EDD  FD 21 13 2F  LD     IY,2F13      ca. 3 125 000 = 1E8/32
2EE1  CD A0 35     CALL   35A0         mit Argument vergleichen
2EE4  28 1B        JR     Z,2F01       gleich? dann weiter
2EE6  30 08        JR     NC,2EF0      Zahl größer? dann <1E9 machen
2EE8  CD 12 34     CALL   3412         sonst mit 10 multiplizieren
2EEB  D1           POP    DE
2EEC  1D           DEC    E            Kommastellung
2EED  D5           PUSH   DE
2EEE  18 ED        JR     2EDD         nochmals, bis Zahl>=3125000
2EF0  FD 21 18 2F  LD     IY,2F18      1E9
2EF4  CD A0 35     CALL   35A0         mit Zahl vergleichen
2EF7  38 08        JR     C,2F01       Zahl kleiner? dann weiter
2EF9  CD 9B 34     CALL   349B         sonst Zahl durch 10 teilen
2EFC  D1           POP    DE
2EFD  1C           INC    E            Kommastellung
2EFE  D5           PUSH   DE
2EFF  18 EF        JR     2EF0         nochmals, bis Zahl<1E9
2F01  CD 8E 2E     CALL   2E8E         Zahl runden
2F04  79           LD     A,C          Anz. signif. Mantissenbytes
2F05  D1           POP    DE
2F06  C1           POP    BC
2F07  4F           LD     C,A          nach C
2F08  3D           DEC    A            -1
2F09  85           ADD    L              zum Zeiger
2F0A  6F           LD     L,A            auf Zahl addieren,
2F0B  D0           RET    NC             = Zeiger auf Restmantisse
2F0C  24           INC    H
2F0D  C9           RET
2F0E  5F           LD     E,A          Kommastellung:=0
2F0F  77           LD     (HL),A       Null als Zahl setzen
2F10  0E 01        LD     C,01         1 signif. Mantissenbyte
2F12  C9           RET

2F13  F0 1F BC 3E 96                   3 124 999.98, ca. 1E8/32
2F18  FE 27 6B 6E 9E                   1E9
```

```
**********************************       FLO Zahl mit 10^A multiplizieren
                                         IN : A: Zehnerexponent
                                              (HL): FLO-Zahl
                                         OUT: (HL): Zahl*10^A
2F1D  2F          CPL                         Zweierkomplement
2F1E  3C          INC    A                    des 10er-Exp. bilden
2F1F  B7          OR     A                 Null?
2F20  37          SCF                      CY:=1 f. o.k.
2F21  C8          RET    Z                 A=0? dann 10^A=1, raus
2F22  4F          LD     C,A               -Exp. retten
2F23  F2 28 2F    JP     P,2F28            -Exp. positiv(Exp. negativ)?
2F26  2F          CPL                         sonst erneut
2F27  3C          INC    A                    Zweierkomplement bilden
2F28  CD 3E 2F    CALL   2F3E              Restexp. u. Zehnerpot. holen
2F2B  28 09       JR     Z,2F36            Restexp=0? d. letzte Multipl.
2F2D  C5          PUSH   BC
2F2E  F5          PUSH   AF
2F2F  CD 36 2F    CALL   2F36              10er-Pot. m. lfd. Zahl multip.
2F32  F1          POP    AF
2F33  C1          POP    BC
2F34  18 F2       JR     2F28              und nächste Zehnerpotenz
2F36  79          LD     A,C               -Exp
2F37  B7          OR     A                 positiv? (Exp negativ?)
2F38  F2 9E 34    JP     P,349E            dann lfd. Erg./Zehnerpot.
2F3B  C3 15 34    JP     3415              sonst lfd. Erg.*Zehnerpot.

**********************************       Restexp. u. Zehnerpot. holen
                                         IN : A: Exponent
                                         OUT: A: Restexponent
                                              (DE): Zehnerpot. (max 10^13)
                                              Z=1, wenn kein Restexp.
2F3E  11 8F 2F    LD     DE,2F8F           10^13
2F41  D6 0D       SUB    0D                Exp. -13
2F43  D0          RET    NC                Exp>=13? dann zurück
2F44  C6 0C       ADD    0C                sonst Exp-1
2F46  5F          LD     E,A               nach E
2F47  87          ADD    A                    Exponent mit 5
2F48  87          ADD    A                    multiplizieren f. 5 Bytes
2F49  83          ADD    E                    pro FLO-Zahl
2F4A  C6 53       ADD    53                   und als Index zum
2F4C  5F          LD     E,A                  Tabellenstart
2F4D  CE 2F       ADC    2F                   addieren, ergibt
2F4F  93          SUB    E                    Zeiger auf entspr.
2F50  57          LD     D,A                  Zehnerpotenz
2F51  AF          XOR    A                 Z:=1 f. kein Restexponent
2F52  C9          RET

2F53  00 00 00 20 84                       10
2F58  00 00 00 48 87                       100
2F5D  00 00 00 7A 8A                       1000
2F62  00 00 40 1C 8E                       10000
2F67  00 00 50 43 91                       100000
2F6C  00 00 24 74 94                       1000000
2F71  00 80 96 18 98                       10000000
2F76  00 20 BC 3E 9B                       100000000
2F7B  00 28 6B 6E 9E                       1E+09
2F80  00 F9 02 15 A2                       1E+10
2F85  40 B7 43 3A A5                       1E+11
```

```
2F8A  10 A5 D4 68 A8                       1E+12
2F8F  2A E7 84 11 AC                       1E+13

***********************************   FLO RND INITIALIZE
2F94  21 65 89      LD     HL,8965          Ausgangswert
2F97  22 E6 B8      LD     (B8E6),HL        für
2F9A  21 07 6C      LD     HL,6C07          RND
2F9D  22 E4 B8      LD     (B8E4),HL        setzen
2FA0  C9            RET

***********************************   FLO RND SEED
                                      IN : (HL): FLO-Zahl
                                      OUT: (DE): Zahl xor Wurzel
2FA1  EB            EX     DE,HL         Zeiger auf Zahl nach DE retten
2FA2  CD 94 2F      CALL   2F94          Ausgangswert setzen
2FA5  EB            EX     DE,HL         Zeiger auf Zahl wieder nach HL
2FA6  CD E8 35      CALL   35E8          Vorzeichen der Zahl holen
2FA9  C8            RET    Z             Zahl=0? dann raus
2FAA  11 E4 B8      LD     DE,B8E4       Zeiger auf Wurzel f. RND
2FAD  06 04         LD     B,04          4 Mantissenbytes
2FAF  1A            LD     A,(DE)          Wurzel byteweise
2FB0  AE            XOR    (HL)            mit Mantisse der Zahl
2FB1  12            LD     (DE),A          ex-oderieren und als
2FB2  13            INC    DE              neuen RND-Wert
2FB3  23            INC    HL              setzen
2FB4  10 F9         DJNZ   2FAF
2FB6  C9            RET

***********************************   FLO RND-Funktion
                                      IN : HL: Rückgabeadresse
                                      OUT: (HL): RND-Wert
2FB7  E5            PUSH   HL            Zeiger retten
2FB8  2A E6 B8      LD     HL,(B8E6)     unteres RND-Word
2FBB  01 07 6C      LD     BC,6C07       Wurzel f. oberes Word
2FBE  CD FA 2F      CALL   2FFA          1. Word neu daraus generieren
2FC1  E5            PUSH   HL            und retten
2FC2  2A E4 B8      LD     HL,(B8E4)     oberes RND-Word
2FC5  01 65 89      LD     BC,8965       Wurzel f. unteres Word
2FC8  CD FA 2F      CALL   2FFA          2. Word neu daraus generieren
2FCB  D5            PUSH   DE            Anz. der Überträge retten
2FCC  E5            PUSH   HL            2. Word retten
2FCD  2A E6 B8      LD     HL,(B8E6)     unteres RND-Word
2FD0  CD FA 2F      CALL   2FFA          3. Word neu generieren
2FD3  E3            EX     (SP),HL       und gegen 2. Word vertauschen
2FD4  09            ADD    HL,BC         untere Wurzel addieren
2FD5  22 E4 B8      LD     (B8E4),HL     und als neues oberes RND-Word
2FD8  E1            POP    HL            3. Word
2FD9  01 07 6C      LD     BC,6C07       Wurzel f. oberes Word
2FDC  ED 4A         ADC    HL,BC         addieren
2FDE  C1            POP    BC            dazu noch Anz. d. Überäge
2FDF  09            ADD    HL,BC         addieren
2FE0  C1            POP    BC            und 1. Word
2FE1  09            ADD    HL,BC         addieren
2FE2  22 E6 B8      LD     (B8E6),HL     Erg. als unteres Word setzen
2FE5  E1            POP    HL            Rückgabeadresse
```

```
**********************************   FLO letzter RND-Wert
                                     IN : HL: Rückgabeadresse
                                     OUT: (HL): FLO-RND-Wert
2FE6  E5           PUSH   HL            Rückgabeadresse
2FE7  DD E1        POP    IX            nach IX
2FE9  2A E4 B8     LD     HL,(B8E4)     unteres Word
2FEC  ED 5B E6 B8  LD     DE,(B8E6)     oberes Word
2FF0  01 00 00     LD     BC,0000       Rundungsbyte:=0
2FF3  DD 36 04 80  LD     (IX+04),80    Exp. f. Wert zw. 0.5 u. 1
2FF7  C3 B1 36     JP     36B1          RND-Wert normalisieren

**********************************   neues RND-Word generieren
                                     IN : HL: altes Word
                                          BC: 2. Parameter
                                     OUT: HL: neues Word
                                          DE: Anz. d. Überträge
2FFA  EB           EX     DE,HL         Word nach DE
2FFB  21 00 00     LD     HL,0000       Anz. d. Überträge :=0
2FFE  3E 11        LD     A,11          Zähler
3000  3D           DEC    A             Zähler herunterzählen
3001  C8           RET    Z             schon null? dann raus
3002  29           ADD    HL,HL         linksverschieben
3003  CB 13        RL     E               links-
3005  CB 12        RL     D               rotieren
3007  30 F7        JR     NC,3000       CY=0? dann nächstes Bit
3009  09           ADD    HL,BC         bei Übertrag 2. Param. addier.
300A  30 F4        JR     NC,3000       CY=0? dann nächstes Bit
300C  13           INC    DE            sonst Zähler erhöhen
300D  18 F1        JR     3000          und nächstes Bit

**********************************   FLO LOG10-Funktion
                                     IN : (HL): FLO-Zahl ARGument
                                     OUT: (HL): LOG10(ARG)
300F  11 8B 30     LD     DE,308B       Konstante lg2
3012  18 03        JR     3017          allg. Logarithmieren

**********************************   FLO LOG-Funktion
                                     IN : (HL): FLO-Zahl ARGument
                                     OUT: (HL): LOG(ARG)
3014  11 86 30     LD     DE,3086       Konstante ln2

**********************************   FLO LOGARITHMUS
                                     IN : (HL): Numerus
                                          (DE): 1/(lb Basis)
                                     OUT: (HL): BasisLOG Numerus
3017  CD E8 35     CALL   35E8          Vorzeichen d. Arg
301A  3D           DEC    A             -1
301B  FE 01        CP     01            CY:=1 f. Arg>0
301D  D0           RET    NC            Arg<=0? dann raus, Fehler
301E  D5           PUSH   DE            Zeiger auf 1/lbBasis retten
301F  CD 6C 35     CALL   356C          Exp von Arg holen
3022  F5           PUSH   AF            und retten
3023  DD 36 04 80  LD     (IX+04),80    Arg zwischen 0.5 und 1 bringen
3027  11 81 30     LD     DE,3081       Konstante SQR(.5)
302A  CD 9A 35     CALL   359A          mit Arg vergleichen
302D  30 06        JR     NC,3035       Arg>=SQR(.5)? dann o.k.
302F  DD 34 04     INC    (IX+04)       Arg*2, SQR(.5)<=Arg<SQR(2)
3032  F1           POP    AF            zum Ausgleich alten Exponenten
```

```
3033  3D          DEC    A           erniedrigen
3034  F5          PUSH   AF          und wieder retten
3035  CD 16 33    CALL   3316        Arg -> FAC3
3038  D5          PUSH   DE          Zeiger auf altes Arg
3039  11 32 33    LD     DE,3332     Konstante 1
303C  CD 3F 33    CALL   333F        addieren
303F  EB          EX     DE,HL       FAC3 -> (DE)
3040  E1          POP    HL          Arg -> (HL)
3041  D5          PUSH   DE          Zeiger auf FAC3 retten
3042  11 32 33    LD     DE,3332     Konstante 1
3045  CD 37 33    CALL   3337        von normalisiertem Arg subtr.
3048  D1          POP    DE          norm. Arg+1 in FAC3 -> (DE)
3049  CD 9E 34    CALL   349E        (Arg'-1)/(Arg'+1)
304C  CD A9 32    CALL   32A9        quadrier. u. in Polynomber.

*********************************** Konstanten f. LOGARITHMUS
                                   (Taylor'sche Reihe)
304F  04                             Anz. d. Koeffiz. f. Polynom
3050  4C 4B 57 5E 7F                 .434259751=ca.2/ln2/7
3055  0D 08 9B 13 80                 .576584342=ca.2/ln2/5
305A  23 93 38 76 80                 .961800762=ca.2/ln2/3
305F  20 3B AA 38 82                 2.88539007=   2/ln2/1

3064  D5          PUSH   DE          Zeiger auf Eingangsargument
3065  CD 15 34    CALL   3415        mit Ergebnis multiplizieren
3068  D1          POP    DE          Zeiger auf Arg f. Polynomber.
3069  E3          EX     (SP),HL     bisher. Ergebnis retten
306A  7C          LD     A,H         urspr. Exponent
306B  B7          OR     A           positiv?
306C  F2 71 30    JP     P,3071      dann o.k.
306F  2F          CPL                  sonst 2er Komplement
3070  3C          INC    A             bilden
3071  6F          LD     L,A         und positiven Exp ins Lo-Byte
3072  7C          LD     A,H         urspr. Exp retten
3073  26 00       LD     H,00        und Null ins Hi-Byte
3075  CD 29 2E    CALL   2E29        Exp nach Real bei DE wandeln
3078  EB          EX     DE,HL       FLO-Exp -> (DE)
3079  E1          POP    HL          bisher. Ergebnis
307A  CD 3F 33    CALL   333F        dazu addieren
307D  D1          POP    DE          Zeiger auf 1/lb Basis
307E  C3 15 34    JP     3415        Ergebnis durch lb Basis teilen

*********************************** diverse FLO-Konstanten
3081  34 F3 04 35 80                 .707106781=SQR(.5)
3086  F8 17 72 31 80                 .693147181=ln2
308B  85 9A 20 1A 7F                 .301029996=lg2

*********************************** FLO EXP-Funktion
                                   IN : (HL): FLO-Zahl
                                   OUT: (HL): EXP(Arg)
3090  06 E1       LD     B,E1          Exponenten mit -$1F
3092  CD 07 33    CALL   3307          vergleichen
3095  D2 28 33    JP     NC,3328       Exp kleiner? dann EXP(Arg)=1
3098  11 00 31    LD     DE,3100       max. Wert f. EXP
309B  CD 9A 35    CALL   359A          Vergleich Arg - max Arg
309E  F2 EC 36    JP     P,36EC        Arg>max Arg? dann Überlauf
30A1  11 05 31    LD     DE,3105       min. Arg f. EXP
30A4  CD 9A 35    CALL   359A          mit Arg vergleichen
```

```
30A7  FA E6 36     JP    M,36E6       Arg<min Arg? dann Unterlauf
30AA  11 FB 30     LD    DE,30FB      Konstante 1/ln2
30AD  CD D4 32     CALL  32D4         mal Arg, Vor- u. Nachkommast.
30B0  7B           LD    A,E          ganzzahliger Teil
30B1  F2 B6 30     JP    P,30B6       Exp positiv?
30B4  ED 44        NEG                sonst 2er Komplement bilden
30B6  F5           PUSH  AF           positiven Exponenten retten
30B7  CD 1D 33     CALL  331D         Exp quadrieren
30BA  CD 0F 33     CALL  330F         und Ergebnis -> FAC1
30BD  D5           PUSH  DE           Zeiger auf Ergebnis retten
30BE  CD AC 32     CALL  32AC         1. Polynom in FAC1 berechnen

**********************************  Konstanten f. EXP, 1. Polynom
30C1  03                              Anz. der Konstanten
30C2  F4 32 EB 0F 73                  6.8625806E-05 = 1/e^(12-3)/2
30C7  08 B8 D5 52 7B                  .0257366747   = 1/e^( 6-3)/2
30CC  00 00 00 00 80                  .5            = 1/e^( 3-3)/2

30D1  E3           EX    (SP),HL      Zeiger Arg' nach HL
30D2  CD AC 32     CALL  32AC         2. Polynom in (HL) berechnen

**********************************  Konstanten f. EXP, 2. Polynom
30D5  02
30D6  09 60 DE 01 78                  1.98163838E-03 = 1/(5*ln2)^5
30DB  F8 17 72 31 7E                  .173286795     = ln2/4

30E0  CD 15 34     CALL  3415         Ergebnis mal Arg'
30E3  D1           POP   DE           1. Polynom -> (DE)
30E4  E5           PUSH  HL           2. Polynom retten
30E5  EB           EX    DE,HL
30E6  CD 37 33     CALL  3337         1. Polynom-2. Polynom -> FAC1
30E9  EB           EX    DE,HL        FAC1 -> (DE)
30EA  E1           POP   HL           2. Polynom -> (HL)
30EB  CD 9E 34     CALL  349E         2.Polyn./(1.Polyn.-2.Polyn.)
30EE  11 CC 30     LD    DE,30CC      0.5
30F1  CD 3F 33     CALL  333F         addieren
30F4  DD 34 04     INC   (IX+04)      *2
30F7  F1           POP   AF             mit 2^(ganzzahl. Teil)
30F8  C3 7B 35     JP    357B           multiplizieren

**********************************  diverse Konstanten f. EXP
30FB  29 3B AA 38 81                    1.44269504 = 1/ln(2)
3100  C7 33 0F 30 87                    88.0296919 = max. Arg. f. EXP
3105  F8 17 72 B1 87                  -88.7228392 = min. Arg. f. EXP

**********************************  FLO SQR-Funktion
                                     IN : (HL): FLO-Zahl
                                     OUT: (HL): SQR(Arg)
310A  11 CC 30     LD    DE,30CC        0.5 als Exponenten setzen

**********************************  FLO Potenzierung
                                     IN : (HL): FLO-Basis
                                          (DE): FLO-Exponent
                                     OUT: (HL): Basis ^ Exponent
310D  EB           EX    DE,HL
310E  CD E8 35     CALL  35E8           Vorz. d. Exponenten
3111  EB           EX    DE,HL
3112  CA 28 33     JP    Z,3328         Exp=0? d. 1 holen, raus
```

```
3115  F5        PUSH  AF        Vorz. d. Exp retten
3116  CD E8 35  CALL  35E8      Vorz. d. Basis holen
3119  28 25     JR    Z,3140    Basis=0? d. entspr. behandeln
311B  47        LD    B,A       Vorz. d. Basis retten
311C  FC FB 35  CALL  M,35FB    neg.? dann invertieren
311F  E5        PUSH  HL
3120  CD 82 31  CALL  3182      Vorz. d. Erg. bestimmen
3123  E1        POP   HL
3124  38 25     JR    C,314B    Integerexp<39? d. entspr. beh.
3126  E3        EX    (SP),HL   sonst Vorz. d. Exp
3127  E1        POP   HL        vom Stack
3128  FA 48 31  JP    M,3148    Erg. neg.? dann Fehler
312B  C5        PUSH  BC
312C  D5        PUSH  DE
312D  CD 14 30  CALL  3014      LOG der Basis bilden
3130  D1        POP   DE        Zeiger auf Exp
3131  DC 15 34  CALL  C,3415    o.k.? d. Exp*LOG(Basis)
3134  DC 90 30  CALL  C,3090    o.k.? d. EXP(Exp*LOG(Basis))
3137  C1        POP   BC        Vorz. d. Ergebnisses
3138  D0        RET   NC        Fehler? dann raus
3139  78        LD    A,B       Vorz. d. Ergebnisses
313A  B7        OR    A         Ergebnis negativ?
313B  FC FB 35  CALL  M,35FB    dann Vorzeichen invertieren
313E  37        SCF             CY:=1 f. o.k.
313F  C9        RET

3140  F1        POP   AF        Vorz. d. Exp
3141  37        SCF             CY:=1 f. o.k.
3142  F0        RET   P         Exp positiv? dann Erg:=0, raus
3143  CD EC 36  CALL  36EC      sonst Überlauf
3146  AF        XOR   A         CY:=0 f. Fehler
3147  C9        RET
3148  AF        XOR   A         CY:=0 f. Fehler
3149  3C        INC   A         A:=1 f. neg. Basis, gebr. Exp
314A  C9        RET

314B  4F        LD    C,A       Exponent retten
314C  F1        POP   AF        Vorzeichen d. Exp
314D  C5        PUSH  BC        Vorz. d. Ergebnisses retten
314E  F5        PUSH  AF        Vorz. d. Exp retten
314F  79        LD    A,C       Exponent
3150  37        SCF             CY:=1 als Markierung
3151  8F        ADC   A         b7 heraus, Marke hinein
3152  30 FD     JR    NC,3151   bis obertstes 1-Bit schieben
3154  47        LD    B,A       Exp retten
3155  CD 0F 33  CALL  330F      Basis -> FAC1
3158  EB        EX    DE,HL     FAC1->(DE),Orig.-Basis->(HL)
3159  78        LD    A,B       Exp wieder zurück
315A  87        ADD   A         b7 ins Carry schieben
315B  28 15     JR    Z,3172    Marke schon draußen? d. Ende
315D  F5        PUSH  AF        Exp u. Flags retten
315E  CD 1D 33  CALL  331D      Basis quadrieren
3161  30 16     JR    NC,3179   Fehler? dann behandeln
3163  F1        POP   AF        Exp u. Flags zurück
3164  30 F4     JR    NC,315A   war b7=0? dann nächstes Bit
3166  F5        PUSH  AF        sonst Exp u. Flags retten
3167  11 E8 B8  LD    DE,B8E8   FAC1, enthält Basis
316A  CD 15 34  CALL  3415      m. lfd. Ergebnis multipliz.
```

```
316D  30 0A        JR     NC,3179      Fehler? dann behandeln
316F  F1           POP    AF           Exp zurück
3170  18 E8        JR     315A         und nächstes Bit bearbeiten
3172  F1           POP    AF           Vorz d. Exp
3173  37           SCF                 CY:=1 f. o.k.
3174  FC FD 32     CALL   M,32FD       Exp negativ? d. Kehrwert
3177  18 BE        JR     3137         raus

3179  F1           POP    AF           Exp vom Stack löschen
317A  F1           POP    AF           Vorz d. Exp
317B  C1           POP    BC           vom Stack entfernen
317C  FA E6 36     JP     M,36E6       Exp neg.? d. Überlauf->Unterl.
317F  C3 EE 36     JP     36EE         sonst Überlauf

*********************************  Vorzeichen d. Erg. best.
                                   IN : B: Vorz. d. Basis
                                        (DE): FLO-Exp
                                   OUT: CY=1, wenn Integerexp<39
                                          A: Integerexp
                                          B: Vorz. d. Ergebnisses
                                        CY=0, sonstige Exp
                                          A=00 f. o.k. (b7=1 Fehler)
                                          B: Vorz. d. Ergebnisses
3182  C5           PUSH   BC           Vorz. d. Basis retten
3183  CD 17 33     CALL   3317         Exp -> FAC3
3186  CD A1 2E     CALL   2EA1         Nachkommastellen abschneiden
3189  79           LD     A,C          Länge der Mantisse
318A  C1           POP    BC           Vorz. d. Basis
318B  30 02        JR     NC,318F      Zahl zu groß? d. behandeln
318D  28 03        JR     Z,3192       Integerexp? d. behandeln
318F  78           LD     A,B          Vorz. d. Erg.:= Vorz. d. Basis
3190  B7           OR     A            Flags entspr. setzen
3191  C9           RET

3192  4F           LD     C,A          Länge der Mantisse retten
3193  7E           LD     A,(HL)       unterstes Mantissenbyte
3194  1F           RRA                 b0 ins Carry schieben
3195  9F           SBC    A            A:=$FF f. b0=1, Exp ungerade
3196  A0           AND    B            m. Vorz. d. Basis undieren
3197  47           LD     B,A          als Ergebnis-Vorzeichen
3198  79           LD     A,C          Länge der Mantisse
3199  FE 02        CP     02           >=2?
319B  9F           SBC    A            dann A:=00,CY:=0 f. Fehler
319C  D0           RET    NC           und raus
319D  7E           LD     A,(HL)       sonst das Mantissenbyte laden
319E  FE 27        CP     27           <39?
31A0  D8           RET    C            dann CY=1, raus
31A1  AF           XOR    A            sonst CY:=0, A:=00 f. o.k.
31A2  C9           RET

*********************************  FLO PI-Funktion
                                   IN : HL: Rückgabeadresse
                                   OUT: (HL): Konstante Pi
31A3  11 A9 31     LD     DE,31A9      Zeiger auf Pi
31A6  C3 18 2E     JP     2E18         FLO-Zahl nach HL kopieren

*********************************  Konstante Pi
31A9  A2 DA 0F 49 82                   FLO-Konstante Pi
```

```
*********************************  FLO DEG/RAD
                                   IN : A: $00 f. RAD
                                           $FF f. DEG
31AE  32 F7 B8     LD     (B8F7),A    Flag setzen
31B1  C9           RET

*********************************  FLO COS-Funktion
                                   IN : (HL): FLO-Zahl
                                   OUT: (HL): COS(Arg)
31B2  CD E8 35     CALL   35E8        Vorzeichen d. Arg
31B5  FC FB 35     CALL   M,35FB      negativ? d. Vorz. invert.
31B8  F6 01        OR     01          A:<>0, Flag f. COS
31BA  18 01        JR     31BD        allg. Routine SIN/COS

*********************************  FLO SIN-Funktion
                                   IN : (HL): FLO-Zahl
                                   OUT: (HL): SIN(Arg)
31BC  AF           XOR    A           A:=0, Flag f. SIN

*********************************  allg. SIN/COS-Funktion
                                   IN : A: 0 f. SIN
                                           sonst COS
                                        (HL): FLO-Zahl
                                   OUT: (HL): SIN/COS(Arg)
31BD  F5           PUSH   AF          SIN/COS-Flag retten
31BE  11 1D 32     LD     DE,321D     Konstante 1/Pi f. RAD
31C1  06 F0        LD     B,F0        Näherungslimit-Exp -$10 f. RAD
31C3  3A F7 B8     LD     A,(B8F7)    DEG-Flag
31C6  B7           OR     A           RAD eingestellt?
31C7  28 05        JR     Z,31CE      dann 1/Pi behalten
31C9  11 22 32     LD     DE,3222     sonst Zeiger auf Konst. 1/180
31CC  06 F6        LD     B,F6        und neuer Grenzexp: -$0A, DEG
31CE  CD 07 33     CALL   3307        Exponenten vergleichen
31D1  30 3A        JR     NC,320D     Exp<Grenzexp? d. SIN(Arg):=Arg
31D3  F1           POP    AF          SIN/COS-Flag
31D4  CD D5 32     CALL   32D5        Arg entspr. normalisieren
31D7  D0           RET    NC          Fehler? dann raus
31D8  7B           LD     A,E         Integerwert
31D9  1F           RRA                ungerade? (negative Periode)
31DA  DC FB 35     CALL   C,35FB      dann Vorzeichen umdrehen
31DD  06 E8        LD     B,E8        Vergleichsexp -$18
31DF  CD 07 33     CALL   3307        mit Arg Exp vergleichen
31E2  D2 E6 36     JP     NC,36E6     Arg Exp kleiner? dann Erg.:=0
31E5  DD 34 04     INC    (IX+04)     sonst Arg'*2
31E8  CD A9 32     CALL   32A9        Polynomberechnung

*********************************  Konstanten für SIN/COS
                                   (MacLaurin'sche Reihe)
31EB  06                              Anzahl der Konstanten
31EC  1B 2D 1A E6 6E                 -3.42879073E-06 = -(Pi/2)^11/11!
31F1  F8 FB 07 28 74                  1.60247029E-04 =  (Pi/2)^ 9/ 9!
31F6  01 89 68 99 79                 -4.68165102E-03 = -(Pi/2)^ 7/ 7!
31FB  E1 DF 35 23 7D                  .0796926013    =  (Pi/2)^ 5/ 5!
3200  28 E7 5D A5 80                 -.645964095     = -(Pi/2)^ 3/ 3!
3205  A2 DA 0F 49 81                  1.57079633     =  (Pi/2)^ 1/ 1!

320A  C3 15 34     JP     3415        Erg. nochmals mit Arg' mult.
```

```
320D  F1            POP   AF          SIN/COS-Flag
320E  C2 28 33      JP    NZ,3328     COS? dann 1 als Näherung
3211  3A F7 B8      LD    A,(B8F7)    DEG-Flag
3214  FE 01         CP    01          RAD?
3216  D8            RET   C           dann Arg als Näherung
3217  11 27 32      LD    DE,3227     sonst mit Pi/180
321A  C3 15 34      JP    3415        mult., Arg*Pi/180 als Näherung

*********************************** div. Konstanten f. SIN/COS
321D  6E 83 F9 22 7F                    .318309886     = 1/Pi
3222  B6 60 0B 36 79                    5.55555555E-03 = 1/180
3227  13 35 FA 0E 7B                    .0174532925    = Pi/180
322C  D3 E0 2E 65 86                    57.2957795     = 180/Pi

*********************************** FLO TAN-Funktion
                                  IN : (HL): FLO-Zahl
                                  OUT: (HL): TAN(Arg)
3231  CD 0F 33      CALL  330F          Arg nach FAC1 kopieren
3234  D5            PUSH  DE            Zeiger auf Zahl retten
3235  CD B2 31      CALL  31B2          COS berechnen
3238  E3            EX    (SP),HL       COS(Arg) auf Stack, Arg->(HL)
3239  DC BC 31      CALL  C,31BC        kein Fehler? d. SIN(Arg) ber.
323C  D1            POP   DE            Zeiger auf COS(Arg)
323D  DA 9E 34      JP    C,349E        o.k.? d. SIN(Arg)/COS(Arg)
3240  C9            RET

*********************************** FLO ATN-Funktion
                                  IN : (HL): FLO-Zahl
                                  OUT: (HL): ATN(Arg)
3241  CD E8 35      CALL  35E8          Vorzeichen des Arg
3244  F5            PUSH  AF            retten
3245  FC FB 35      CALL  M,35FB        und, wenn neg., invertieren
3248  06 F0         LD    B,F0          Grenzexp -$10
324A  CD 07 33      CALL  3307          mit Exp v. Arg vergleichen
324D  30 4A         JR    NC,3299       Exp Arg<Exp? d. ATN(Arg):=Arg
324F  3D            DEC   A             Exp d. Arg -1
3250  F5            PUSH  AF            retten
3251  F4 FD 32      CALL  P,32FD        wenn pos. (Arg>1) Kehrwert
3254  CD A9 32      CALL  32A9          Arg quadrieren, Polynomber.

*********************************** Konstanten f. ATN
                                  (für Taylor'sche Reihe)
3257  0B                                Anz. d. Konstanten
3258  FF C1 03 0F 77                    1.09111541E-03 = ca.  1/21
325D  83 FC E8 EB 79                   -7.199405E-03   = ca. -1/19
3262  6F CA 78 36 7B                    .0222743944    = ca.  1/17
3267  D5 3E B0 B5 7C                   -.0443575339    = ca. -1/15
326C  B0 C1 8B 09 7D                    .0671610958    = ca.  1/13
3271  AF E8 32 B4 7D                   -.0879877261    = ca. -1/11
3276  74 6C 65 62 7D                    .110545013     =  1/ 9
327B  D1 F5 37 92 7E                   -.142791596     = -1/ 7
3280  7A C3 CB 4C 7E                    .199996046     =  1/ 5
3285  83 A7 AA AA 7F                   -.333333239     = -1/ 3
328A  FE FF FF 7F 80                    1              =  1/ 1

328F  CD 15 34      CALL  3415          Polynom noch mit Arg' multip.
3292  F1            POP   AF            Exp d. Arg -1
3293  11 05 32      LD    DE,3205       Konstante Pi/2
```

```
3296  F4 3B 33     CALL   P,333B       Erg.:=Erg.-Pi/2, wenn Kehrwert
3299  3A F7 B8     LD     A,(B8F7)     DEG-Flag
329C  B7           OR     A            testen
329D  11 2C 32     LD     DE,322C      f. DEG m. 180/Pi
32A0  C4 15 34     CALL   NZ,3415      multiplizieren
32A3  F1           POP    AF           Vorz. d. Arg
32A4  FC FB 35     CALL   M,35FB       Erg., wenn Arg neg., invert.
32A7  37           SCF                 CY:=1 f. o.k.
32A8  C9           RET

**********************************  Polynomberechnung mit Arg^2
                                    IN : (HL): FLO-Zahl
                                         (SP): Zeiger auf Tabelle
                                    OUT: (HL): P(Arg^2)
                                         (DE): Arg
                                         (SP): Zeiger hinter Tabelle
32A9  CD 1D 33     CALL   331D           Arg quadrieren

**********************************  Polynomberechnung
                                 P(x):=(((a1x+a2)x+a3)x+a4)x+...
                                    IN : (HL): FLO-Zahl, x
                                         (SP): Zeiger auf Tabelle mit
                                               den Koeffiz. a1 .. ai
                                    OUT: (HL): P(Arg)
                                         (DE): Arg in FAC2
                                         (SP): Zeiger hinter Tabelle
32AC  CD 16 33     CALL   3316         Zahl nach FAC3 kopieren
32AF  EB           EX     DE,HL        Zeiger Arg -> HL
32B0  D1           POP    DE           Anfangsadresse der Tabelle
32B1  1A           LD     A,(DE)       Anz. d. Koeffizienten
32B2  13           INC    DE           Zeiger auf ersten Koeffiz.
32B3  47           LD     B,A          Anzahl nach B
32B4  CD 18 2E     CALL   2E18         Koeffiz. aus Tabelle kopieren
32B7  13           INC    DE             Zeiger
32B8  13           INC    DE             auf
32B9  13           INC    DE             nächsten
32BA  13           INC    DE             Koeffizienten
32BB  13           INC    DE             aus Tabelle
32BC  D5           PUSH   DE           als evtl. Rückkehradr. pushen
32BD  11 ED B8     LD     DE,B8ED      Zeiger auf FAC2 (Arg)
32C0  05           DEC    B            Anz. d. Koeffiz. dekr.
32C1  C8           RET    Z            null? dann zurück, hinter Tab.
32C2  C5           PUSH   BC           Zähler retten
32C3  11 F2 B8     LD     DE,B8F2      Zeiger auf FAC3 (Arg)
32C6  CD 15 34     CALL   3415         lfd. Erg. damit multipliz.
32C9  C1           POP    BC           Zähler
32CA  D1           POP    DE           Zeiger auf Koeffiz. in Tabelle
32CB  D5           PUSH   DE           wieder retten
32CC  C5           PUSH   BC           wieder retten
32CD  CD 3F 33     CALL   333F         nächsten Koeffizienten add.
32D0  C1           POP    BC           Zähler
32D1  D1           POP    DE           Zeiger in Tabelle
32D2  18 E3        JR     32B7         nächsten Koeffizienten bearb.
```

```
**********************************    Arg normalis. f. EXP
                                      IN : (HL): Argument, FLO
                                           (DE): Faktor
                                      OUT: (HL): normal. Argument
                                            DE: Integerwert Arg
                                            AF: Flags und Vorz. f. Arg
32D4  AF          XOR    A

**********************************    Arg normalis. f. SIN/COS
                                      IN : (HL): Argument, FLO
                                           (DE): Faktor
                                           Z=0 f. .5 addieren
                                      OUT: (HL): normal. Argument
                                            DE: Integerwert Arg
                                            AF: Flags und Vorz. f. Arg
32D5  F5          PUSH   AF             Flag retten
32D6  CD 15 34    CALL   3415           Arg m. Faktor multiplizieren
32D9  F1          POP    AF             Flag wiederholen
32DA  11 CC 30    LD     DE,30CC        Konstante 0.5
32DD  C4 3F 33    CALL   NZ,333F        ggf. add.,Phasenversch. ausgl.
32E0  E5          PUSH   HL             Ergebnis retten
32E1  CD 66 2E    CALL   2E66           und nach Integer wandeln
32E4  30 13       JR     NC,32F9        Fehler? dann raus
32E6  D1          POP    DE             Zeiger auf verändertes Arg
32E7  E5          PUSH   HL             Integer retten
32E8  F5          PUSH   AF             Flags u. Vorz. retten
32E9  D5          PUSH   DE             Zeiger auf Arg retten
32EA  11 ED B8    LD     DE,B8ED        Zeiger auf FAC2
32ED  CD 29 2E    CALL   2E29           Integer nach Real wandeln
32F0  EB          EX     DE,HL          und FLO-Integerteil -> (DE)
32F1  E1          POP    HL             Zeiger auf Arg
32F2  CD 37 33    CALL   3337           Arg-INT(Arg)=Nachkommateil
32F5  F1          POP    AF             Flags u. Vorz.
32F6  D1          POP    DE             Integerteil von Arg
32F7  37          SCF                   CY:=1 f. o.k.
32F8  C9          RET

32F9  E1          POP    HL             Zeiger auf Arg
32FA  AF          XOR    A              CY:=0 f. Fehler
32FB  3C          INC    A              Z:=0
32FC  C9          RET

**********************************    FLO Kehrwert bilden
                                      IN : (HL): FLO-Zahl
                                      OUT: (HL): 1/Arg
32FD  CD 16 33    CALL   3316           Zahl in FAC3 schreiben
3300  EB          EX     DE,HL
3301  CD 28 33    CALL   3328           Konstante 1 holen
3304  C3 9E 34    JP     349E           und durch Arg teilen

**********************************    Exponenten vergleichen
                                      IN : B: Vergleichsexp
                                           (HL): FLO-Zahl
                                      OUT: A: Exp, 2er Kompl.
                                           CY=0, wenn Exp<B
3307  CD 6C 35    CALL   356C           Exp im 2er Kompl. holen
330A  F0          RET    P              Arg>=1? dann zurück
330B  B8          CP     B              Exp u. Verlechsexp vergleichen
```

```
330C  C8           RET    Z          gleich? dann zurück
330D  3F           CCF               sonst CY:=0 f. Exp<B
330E  C9           RET

**********************************  Argument nach FAC1 kopieren
                                    IN : (HL): Argument (5 Bytes)
                                    OUT: Argument in FAC1
                                         HL: Zeiger auf FAC1
330F  EB           EX     DE,HL
3310  21 E8 B8     LD     HL,B8E8      Zeiger auf FAC1
3313  C3 18 2E     JP     2E18         Zahl dorthin kopieren

**********************************  Argument nach FAC3 kopieren
                                    IN : (HL): Argument (5 Bytes)
                                    OUT: Argument in FAC3
3316  EB           EX     DE,HL
3317  21 F2 B8     LD     HL,B8F2      Zeiger auf FAC3
331A  C3 18 2E     JP     2E18         Zahl dorthin kopieren

**********************************  Zahl quadrieren
                                    IN : (HL): FLO-Zahl
                                    OUT: (HL): Arg^2
331D  EB           EX     DE,HL        Zahl
331E  21 ED B8     LD     HL,B8ED      nach FAC2
3321  CD 18 2E     CALL   2E18         kopieren
3324  EB           EX     DE,HL        und mit diesem
3325  C3 15 34     JP     3415         multiplizieren

**********************************  Konstante 1 holen
                                    IN : (HL): Rückgabeadresse
                                    OUT: (HL): FLO 1
3328  D5           PUSH   DE
3329  11 32 33     LD     DE,3332      Zeiger auf FLO-Konstante 1
332C  CD 18 2E     CALL   2E18         nach HL kopieren
332F  D1           POP    DE
3330  37           SCF                 CY:=1 f. o.k.
3331  C9           RET

**********************************  Konstante 1
3332  00 00 00 00 81

**********************************  FLO (HL):=(HL)-(DE)
                                    IN : (HL): FLO-Zahl
                                         (DE): FLO-Zahl
                                    OUT: (HL): Differenz
3337  3E 01        LD     A,01         b0:=1 f. (DE) invertieren
3339  18 05        JR     3340         Addition

**********************************  FLO (HL):=(DE)-(HL)
                                    IN : (HL): FLO-Zahl
                                         (DE): FLO-Zahl
                                    OUT: (HL): Differenz
333B  3E 80        LD     A,80         b7:=1 f. (HL) negieren
333D  18 01        JR     3340         Addition

**********************************  FLO (HL):=(HL)+(DE)
                                    IN : (HL): FLO-Zahl
                                         (DE): FLO-Zahl
```

```
                                   OUT: (HL): Summe
333F  AF          XOR    A            b0=b7=0 f. beide so lassen

*********************************  allg. Addition/Subtraktion
                                   IN : (HL): FLO-Zahl
                                        (DE): FLO-Zahl
                                        A<b0>:Flag f. -(DE)
                                        A<b7>:FLAG f. -(HL)
                                   OUT: (HL): Summe
3340  E5          PUSH   HL           Zeiger Arg1
3341  DD E1       POP    IX           -> IX
3343  D5          PUSH   DE           Zeiger Arg2
3344  FD E1       POP    IY           -> IY
3346  DD 46 03    LD     B,(IX+03)    Vorzeichen Arg1 -> B
3349  FD 4E 03    LD     C,(IY+03)    Vorzeichen Arg2 -> C
334C  B7          OR     A            Flag f. Vorz. testen
334D  28 0B       JR     Z,335A       keine Vorzeichen invertieren?
334F  FA 58 33    JP     M,3358       Arg1 negieren?
3352  3E 80       LD     A,80         sonst Vorzeichen von
3354  A9          XOR    C            Arg2 invertieren
3355  4F          LD     C,A          und als neues Vorz v. Arg2
3356  18 02       JR     335A         dann normal addieren
3358  A8          XOR    B            Vorzeichen von Arg1 invertier.
3359  47          LD     B,A          und neu setzen

*********************************  Addieren
                                   IN : (HL)=(IX): Arg1
                                        (DE)=(IY): Arg2
                                        B: Vorz. Arg1
                                        C: Vorz. Arg2
                                   OUT: (HL): Arg1+Arg2
335A  DD 7E 04    LD     A,(IX+04)    Exp1
335D  FD BE 04    CP     (IY+04)      mit Exp2 vergleichen
3360  30 14       JR     NC,3376      Exp1>=Exp2? dann entspr. beh.
3362  50          LD     D,B            sonst
3363  41          LD     B,C            Vorzeichen
3364  4A          LD     C,D            vertauschen
3365  B7          OR     A            CY:=0
3366  57          LD     D,A          Exp1 retten
3367  FD 7E 04    LD     A,(IY+04)    Exp2
336A  DD 77 04    LD     (IX+04),A    als Exp1 setzen
336D  28 54       JR     Z,33C3       Exp1=0? d. Erg:=(IY)
336F  92          SUB    D            sonst Exp2-Exp1=Bitdifferenz
3370  FE 21       CP     21           >Länge einer FLO-Zahl?
3372  30 4F       JR     NC,33C3      dann ERG.:=(IY)
3374  18 11       JR     3387         sonst Addition ausführen

3376  AF          XOR    A            Zweierkomplement
3377  FD 96 04    SUB    (IY+04)      von Exp2 bilden
337A  28 59       JR     Z,33D5       Exp2=0? dann Erg:=Arg1, raus
337C  DD 86 04    ADD    (IX+04)      sonst Stellendifferenz berech.
337F  FE 21       CP     21           >Länge einer FLO-Mantisse?
3381  30 52       JR     NC,33D5      dann Erg.:=Arg1, raus
3383  E5          PUSH   HL           Zeiger auf Arg1
3384  FD E1       POP    IY           nach IY
3386  EB          EX     DE,HL        Zeiger Arg2 nach HL

3387  5F          LD     E,A          Anz. d. Bitstellen retten
```

```
3388  78          LD    A,B          Vorzeichen
3389  A9          XOR   C            vergleichen
338A  F5          PUSH  AF           und Vergleichsergebnis retten
338B  C5          PUSH  BC           Vorzeichen retten
338C  7B          LD    A,E          Anz. d. zu shiftenden Bits
338D  CD 43 36    CALL  3643         (HL) rechtsverschieben
3390  79          LD    A,C          Rundungsbyte retten
3391  C1          POP   BC           Vorzeichen
3392  4F          LD    C,A          Rundungsbyte nach C
3393  F1          POP   AF           Vorzeichenvergleich
3394  FA DA 33    JP    M,33DA       Vorz. ungl.? d. Mant. subtrah.

3397  FD 7E 00    LD    A,(IY+00)      bei gleichen Vorzeichen:
339A  85          ADD   L              Mantissen
339B  6F          LD    L,A            addieren
339C  FD 7E 01    LD    A,(IY+01)        (IY+00)..(IY+03)
339F  8C          ADC   H                +   HL  ,   DE
33A0  67          LD    H,A              ->   HL ,  DE
33A1  FD 7E 02    LD    A,(IY+02)
33A4  8B          ADC   E
33A5  5F          LD    E,A
33A6  FD 7E 03    LD    A,(IY+03)
33A9  CB FF       SET   7,A              Vorz. durch 1 ersetzen
33AB  8A          ADC   D
33AC  57          LD    D,A
33AD  D2 BA 36    JP    NC,36BA      kein Übertrag? dann o.k.
33B0  CB 1A       RR    D              sonst
33B2  CB 1B       RR    E              Ergebnismantisse
33B4  CB 1C       RR    H              1 Bit
33B6  CB 1D       RR    L              rechtsverschieben
33B8  CB 19       RR
33BA  DD 34 04    INC   (IX+04)      Exponenten zum Ausgl. erhöhen
33BD  C2 BA 36    JP    NZ,36BA      kein Üertrag? d. runden, raus
33C0  C3 EE 36    JP    36EE         sonst Überlauf

33C3  FD 7E 02    LD    A,(IY+02)      Mantisse von
33C6  DD 77 02    LD    (IX+02),A      (IY)
33C9  FD 7E 01    LD    A,(IY+01)      nach
33CC  DD 77 01    LD    (IX+01),A      (IX)
33CF  FD 7E 00    LD    A,(IY+00)      übertragen
33D2  DD 77 00    LD    (IX+00),A
33D5  DD 70 03    LD    (IX+03),B    Vorz. aus B setzen
33D8  37          SCF                CY:=1 f. o.k.
33D9  C9          RET

33DA  AF          XOR   A            b. ungl. Vorzeichen
33DB  91          SUB   C            2er Komplement d. Rundungsbyt.
33DC  4F          LD    C,A          bilden
33DD  FD 7E 00    LD    A,(IY+00)      und Mantissen
33E0  9D          SBC   L              voneinander abziehen:
33E1  6F          LD    L,A              (IY+00)..(IY+04)
33E2  FD 7E 01    LD    A,(IY+01)        -   HL  ,   DE
33E5  9C          SBC   H                ->   HL ,  DE
33E6  67          LD    H,A
33E7  FD 7E 02    LD    A,(IY+02)
33EA  9B          SBC   E
33EB  5F          LD    E,A
33EC  FD 7E 03    LD    A,(IY+03)
```

```
33EF  CB FF       SET   7,A        Vorz. d. 1 ersetzen
33F1  9A          SBC   D
33F2  57          LD    D,A
33F3  30 16       JR    NC,340B    kein Borger? dann o.k.
33F5  78          LD    A,B          sonst
33F6  2F          CPL                Vorzeichen
33F7  47          LD    B,A          invertieren
33F8  AF          XOR   A          u. Zweierkomplement
33F9  91          SUB   C          des Rundungsbytes
33FA  4F          LD    C,A        bilden
33FB  3E 00       LD    A,00       CY=1
33FD  9D          SBC   L
33FE  6F          LD    L,A          Zeierkomplement
33FF  3E 00       LD    A,00         der Mantisse
3401  9C          SBC   H            bilden
3402  67          LD    H,A
3403  3E 00       LD    A,00
3405  9B          SBC   E
3406  5F          LD    E,A
3407  3E 00       LD    A,00
3409  9A          SBC   D
340A  57          LD    D,A
340B  87          ADD   A          höchstes Bit gesetzt?
340C  DA BA 36    JP    C,36BA     d. runden, m. Vorz. versehen
340F  C3 B1 36    JP    36B1       sonst vorher noch normalis.

*********************************  FLO (HL):=10*(HL)
                                   IN : (HL): FLO Zahl
                                   OUT: (HL):=10*(HL)
3412  11 53 2F    LD    DE,2F53      Zeiger auf Konstante 10

*********************************  FLO (HL):=(HL)*(DE)
                                   IN : (HL): FLO Zahl, Arg1
                                        (DE): FLO Zahl, Arg2
                                   OUT: (HL):=(HL)*(DE)
3415  D5          PUSH  DE           Zeiger Arg2
3416  FD E1       POP   IY           nach IY
3418  E5          PUSH  HL           Zeiger Arg1
3419  DD E1       POP   IX           nach IX
341B  FD 7E 04    LD    A,(IY+04)    Exp2
341E  B7          OR    A            =0?
341F  28 2C       JR    Z,344D       dann Erg.:=0, raus
3421  3D          DEC   A            Exp2-1
3422  CD 48 35    CALL  3548         Vorz. u. Exp d. Erg. berechnen
3425  28 26       JR    Z,344D       Erg.=0? d. Erg.:=0, raus
3427  30 21       JR    NC,344A      Fehler? d. entspr. behandeln
3429  F5          PUSH  AF           Ergebnisexp retten
342A  C5          PUSH  BC           Ergebnisvorz. retten
342B  CD 50 34    CALL  3450         Ergebnismantisse berechnen
342E  79          LD    A,C          Rundungsbyte
342F  C1          POP   BC           Vorzeichen in B
3430  4F          LD    C,A          Rundungsbyte in C
3431  F1          POP   AF           Exp d. Ergebnisses
3432  CB 7A       BIT   7,D          oberstes Mantissenbit=1?
3434  20 0D       JR    NZ,3443      dann Ergebnis abspeichern
3436  3D          DEC   A            sonst Exp erniedrigen
3437  28 14       JR    Z,344D       Null? dann Unterlauf, Erg.:=0
3439  CB 21       SLA   C              Rundungsbyte
```

```
343B  CB 15      RL    L              und
343D  CB 14      RL    H              Mantisse
343F  CB 13      RL    E              linksverschieben
3441  CB 12      RL    D
3443  DD 77 04   LD    (IX+04),A    Exp abspeichern
3446  B7         OR    A            und testen
3447  C2 BA 36   JP    NZ,36BA      <>0? d. runden, Vorz. hinein
344A  C3 EE 36   JP    36EE         sonst Überlauf

344D  C3 E6 36   JP    36E6         Unterlauf, Ergebnis:=0

**********************************  Ergebnismantisse berechnen
                                    IN : (IX): Arg1
                                         (IY): Arg2
                                    OUT: DE,HL Ergebnismantisse
3450  21 00 00   LD    HL,0000        Ergebnismantisse
3453  5D         LD    E,L            auf null
3454  54         LD    D,H            setzen
3455  FD 7E 00   LD    A,(IY+00)    4.MSB v. Arg2
3458  CD 93 34   CALL  3493         mit Arg1-Mantisse multipliz.
345B  FD 7E 01   LD    A,(IY+01)    3.MSB v. Arg2
345E  CD 93 34   CALL  3493         mit Arg1-Mantisse multipliz.
3461  FD 7E 02   LD    A,(IY+02)    2.MSB v. Arg2
3464  CD 93 34   CALL  3493         mit Arg1-Mantisse multipliz.
3467  FD 7E 03   LD    A,(IY+03)    1.MSB v. Arg2
346A  F6 80      OR    80           Vorzeichen durch 1 ersetzen

**********************************  Byte mit FLO-Mantisse multipliz.
                                    IN : A: Byte
                                         (IX): FLO-Mantisse
                                         DE,HL: lfd. Ergebnis
                                    OUT: DE,HL: neue Ergebnismantisse
346C  06 08      LD    B,08         Bitzähler setzen
346E  1F         RRA                unterstes Bit a. Byte rausschi.
346F  4F         LD    C,A          lfd. Byte retten
3470  30 14      JR    NC,3486      b0=0? d. nur n. unten schieb.
3472  7D         LD    A,L            sonst
3473  DD 86 00   ADD   (IX+00)        FLO-Mantisse
3476  6F         LD    L,A            byteweise
3477  7C         LD    A,H            zu lfd. Ergebnismantisse
3478  DD 8E 01   ADC   (IX+01)        addieren
347B  67         LD    H,A
347C  7B         LD    A,E
347D  DD 8E 02   ADC   (IX+02)
3480  5F         LD    E,A
3481  7A         LD    A,D
3482  DD 8E 03   ADC   (IX+03)
3485  57         LD    D,A
3486  CB 1A      RR    D              und
3488  CB 1B      RR    E              (neue) Ergebnismantisse
348A  CB 1C      RR    H              um 1 Bit
348C  CB 1D      RR    L              rechtsverschieben
348E  CB 19      RR    C            incl. Rundungsbyte
3490  10 DE      DJNZ  3470         und das f. alle 8 Bits
3492  C9         RET
```

```
***********************************   Byte mit FLO-Mantisse multipliz.
                                     (m. Test auf Nullbyte v. Multiplik.)
                                      IN : A: Byte
                                           (IX): FLO-Mantisse
                                           DE,HL: lfd. Ergebnis
                                      OUT: DE,HL: neue Ergebnismantisse
3493  B7          OR     A               Byte=0?
3494  20 D6       JR     NZ,346C         sonst Byte bitweise multipliz.
3496  6C          LD     L,H               wenn Byte=0:
3497  63          LD     H,E               lfd. Ergebnismantisse
3498  5A          LD     E,D               gleich alle 8 Bits
3499  57          LD     D,A               rechtsverschieben
349A  C9          RET

***********************************   FLO (HL):=(HL)/10
                                      IN : (HL): FLO Zahl
                                      OUT: (HL):=(HL)/10
349B  11 53 2F    LD     DE,2F53         Zeiger auf Konstante 10

***********************************   FLO (HL):=(HL)/(DE)
                                      IN : (HL): FLO Zahl, Dividend
                                           (DE): FLO Zahl, Divisor
                                      OUT: (HL):=(HL)/(DE)
349E  D5          PUSH   DE              Zeiger auf Arg2
349F  FD E1       POP    IY              nach IY
34A1  E5          PUSH   HL              Zeiger auf Arg1
34A2  DD E1       POP    IX              nach IX
34A4  AF          XOR    A               Zweierkomplement
34A5  FD 96 04    SUB    (IY+04)         d. Exp2 bilden
34A8  28 58       JR     Z,3502          Divisor=0? dann Überlauf
34AA  CD 48 35    CALL   3548            sonst Exp add. u. Vorz. ber.
34AD  CA E6 36    JP     Z,36E6          Erg.=0? dann Erg:=0, raus
34B0  30 4D       JR     NC,34FF         Fehler? dann Überlauf behand.
34B2  C5          PUSH   BC              Vorzeichenvergleich retten
34B3  4F          LD     C,A             Ergebnisexp
34B4  5E          LD     E,(HL)            Mantisse
34B5  23          INC    HL                des Arg1
34B6  56          LD     D,(HL)            (Dividend)
34B7  23          INC    HL                nach
34B8  7E          LD     A,(HL)            DE,HL
34B9  23          INC    HL
34BA  66          LD     H,(HL)
34BB  6F          LD     L,A
34BC  EB          EX     DE,HL
34BD  FD 46 03    LD     B,(IY+03)       Vorzeichen Arg1 nach B
34C0  CB F8       SET    7,B             Vorzeichen durch 1 ersetzen
34C2  CD 32 35    CALL   3532            Mantissen vergleichen
34C5  30 06       JR     NC,34CD         Mant2>Mant1? d. Mant1 linksv.
34C7  79          LD     A,C             sonst Ergebnisexp
34C8  B7          OR     A               <>0?
34C9  20 08       JR     NZ,34D3         dann weitermachen
34CB  18 31       JR     34FE            sonst Überlauf
34CD  0D          DEC    C               Ergebnisexp erniedrigen
34CE  29          ADD    HL,HL             und Dividendenmantisse
34CF  CB 13       RL     E                 um 1 Bit
34D1  CB 12       RL     D                 linksverschieben
34D3  DD 71 04    LD     (IX+04),C       Ergebnisexp setzen
34D6  CD 07 35    CALL   3507              Bytes der
```

```
34D9  DD 71 03    LD    (IX+03),C      Ergebnismantisse
34DC  CD 07 35    CALL  3507           einzeln berechnen
34DF  DD 71 02    LD    (IX+02),C      und danach
34E2  CD 07 35    CALL  3507           zwischenspeichern
34E5  DD 71 01    LD    (IX+01),C
34E8  CD 07 35    CALL  3507
34EB  D4 32 35    CALL  NC,3532      ggf. Mantissen vergleichen
34EE  9F          SBC   A            A:=$FF, wenn rest-Mant1>Mant2
34EF  69          LD    L,C          letztes Byte
34F0  DD 66 01    LD    H,(IX+01)      zwischengespeicherte
34F3  DD 5E 02    LD    E,(IX+02)      Ergebnismantisse
34F6  DD 56 03    LD    D,(IX+03)      zurück nach DE,HL
34F9  C1          POP   BC           Vorzeichen
34FA  4F          LD    C,A          Rundungsbyte
34FB  C3 BA 36    JP    36BA         runden, Vorz. hinein, speich.

34FE  C1          POP   BC           Vorzeichen vom Stack löschen
34FF  C3 EE 36    JP    36EE         Überlauf

3502  CD 94 35    CALL  3594         Überlauf
3505  AF          XOR   A            CY:=0 f. Fehler
3506  C9          RET

*********************************** byteweise dividieren
                                   IN : DE,HL: Restmant. Dividend
                                          (IY): Divisormantisse
                                          B: 1.MSB Divisor, b7=1
                                          CY=0, wenn MSB d. Ergebnis
                                   OUT: C: Ergebnisbyte
                                          DE,HL: neue Restmantisse; CY:=1
3507  0E 01       LD    C,01          Byteendemarkierung
3509  38 08       JR    C,3513        noch nicht MSB d. Ergebnisses?
350B  7A          LD    A,D           höchstes Mantissenbyte Arg1
350C  B8          CP    B             höchstes Mantissenbyte Arg2
350D  3F          CCF
350E  CC 36 35    CALL  Z,3536        gleich? d. restl. Mant. vergl.
3511  30 13       JR    NC,3526       Arg2>Arg1? dann nicht subtrah.
3513  7D          LD    A,L             Divisormantisse
3514  FD 96 00    SUB   (IY+00)         byteweise
3517  6F          LD    L,A             von rest-Dividendenmantisse
3518  7C          LD    A,H             subtrahieren
3519  FD 9E 01    SBC   (IY+01)
351C  67          LD    H,A
351D  7B          LD    A,E
351E  FD 9E 02    SBC   (IY+02)
3521  5F          LD    E,A
3522  7A          LD    A,D
3523  98          SBC   B             1.MSB d. Divisormant. in B
3524  57          LD    D,A
3525  37          SCF                 CY:=1 f. Division erfolgt
3526  CB 11       RL    C             in C hinein, ggf. Marke heraus
3528  9F          SBC   A             A:=$FF, wenn Ende erreicht
3529  29          ADD   HL,HL           Restmantisse
352A  CB 13       RL    E               um 1 Bit
352C  CB 12       RL    D               linksverschieben
352E  3C          INC   A             A:=00, wenn Ende erreicht
352F  20 D8       JR    NZ,3509       sonst noch ein Durchlauf
3531  C9          RET
```

```
**********************************   Mantissenvergleich
                                     IN : DE,HL: Mant1
                                          (IY): Mant2
                                          B: 1.MSB Mant2
                                     OUT: Z=1, wenn gleich
                                          CY=1, wenn Mant1 größer
3532  7A          LD    A,D               Mantissen
3533  B8          CP    B                 byteweise
3534  3F          CCF                     vergleichen,
3535  C0          RET   NZ                das Carry
3536  7B          LD    A,E               jeweils invertieren
3537  FD BE 02    CP    (IY+02)
353A  3F          CCF
353B  C0          RET   NZ
353C  7C          LD    A,H
353D  FD BE 01    CP    (IY+01)
3540  3F          CCF
3541  C0          RET   NZ
3542  7D          LD    A,L
3543  FD BE 00    CP    (IY+00)
3546  3F          CCF
3547  C9          RET

**********************************   Exp addieren, Vorz. berechnen
                                     IN : (IX): FLO Arg1
                                          (IY): FLO Arg2
                                          A: Exp2
                                     OUT: B: Vorzeichenvergleich
                                          A: Exp d. Ergebnisses
                                          CY=0, wenn Fehler
                                            S=1, wenn Überlauf
                                            S=0, wenn Unterlauf
                                          Z=1, wenn Ergebnis=0
3548  4F          LD    C,A             Exp2 retten
3549  DD 7E 03    LD    A,(IX+03)       Vorz. Arg1
354C  FD AE 03    XOR   (IY+03)         mit Vorz. Arg2 verknüpfen
354F  47          LD    B,A             und Ergebnis nach B
3550  DD 7E 04    LD    A,(IX+04)       Exp1
3553  B7          OR    A               =0?
3554  C8          RET   Z               dann Erg.:=0, raus
3555  81          ADD   C               Exp2 addieren
3556  4F          LD    C,A             und Ergebnis nach C
3557  1F          RRA                     b7 xor Carry:
3558  A9          XOR   C                 CY=1, wenn Über- o. Unterl.
3559  79          LD    A,C             Ergebnisexp +$7F
355A  F2 68 35    JP    P,3568          =0? dann Fehler, entspr. beh.
355D  DD CB 03 FE SET   7,(IX+03)       sonst Vorz. durch 1 ersetzen
3561  D6 7F       SUB   7F              Ergebnisexp berechnen
3563  37          SCF                   CY:=1 f. o.k.
3564  C0          RET   NZ              wenn Exp<>0, dann raus
3565  FE 01       CP    01              sonst CY:=1, Z=1 f Erg.:=0
3567  C9          RET                   und raus
3568  B7          OR    A               CY:=0
3569  F8          RET   M               Überlauf? d. CY:=0, S:=1, Z:=0
356A  AF          XOR   A               sonst CY:=0, S:=0, Z:=1
356B  C9          RET                   f. Unterlauf setzen
```

```
*********************************** Exponenten holen
                                    IN : (HL): FLO Zahl
                                    OUT: CY=0, wenn Arg=0
                                         A: Exp im 2er Komplement
356C  E5          PUSH   HL            Zeiger auf Zahl
356D  DD E1       POP    IX            nach IX
356F  DD 7E 04    LD     A,(IX+04)     Exp laden
3572  B7          OR     A             Exp=0? (Zahl=0?)
3573  C8          RET    Z             dann raus
3574  D6 80       SUB    80            Exp in 2er Komplementsdarst.
3576  37          SCF                  CY:=1 f. Zahl<>0
3577  C9          RET

*********************************** FLO (HL):=(HL)*2^A
                                    IN : (HL): FLO Zahl
                                         A: Binärexp
                                    OUT: (HL):=(HL)*2^A
3578  E5          PUSH   HL            Zeiger auf Arg
3579  DD E1       POP    IX            nach IX
357B  B7          OR     A             2er Exp
357C  FA 89 35    JP     M,3589        neg.? dann abziehen
357F  DD 86 04    ADD    (IX+04)       sonst zum Exp addieren
3582  DD 77 04    LD     (IX+04),A
3585  3F          CCF                  C:=0 f. Fehler
3586  D8          RET    C             o.k.? dann raus
3587  18 0B       JR     3594          sonst Überlauf
3589  DD 86 04    ADD    (IX+04)       Exp addieren
358C  38 02       JR     C,3590        kein Borger? dann o.k.
358E  AF          XOR    A             sonst null
358F  37          SCF                  CY:=1 f. o.k.
3590  DD 77 04    LD     (IX+04),A     Null setzen
3593  C9          RET

3594  DD 46 03    LD     B,(IX+03)
3597  CD EE 36    CALL   36EE

*********************************** FLO Vergleich (HL)-(DE)
                                    IN : (HL): FLO Zahl, Arg1
                                         (DE): FLO Zahl, Arg2
                                    OUT: A=$00,Z=1:     Arg1=Arg2
                                         A=$01,Z=0,C=0: Arg1>Arg2
                                         A=$FF,Z=0,C=1: Arg1<Arg2
359A  E5          PUSH   HL            Zeiger auf Arg1
359B  DD E1       POP    IX            nach IX
359D  D5          PUSH   DE            Zeiger auf Arg2
359E  FD E1       POP    IY            nach IY
35A0  DD 7E 04    LD     A,(IX+04)     Exp1
35A3  FD BE 04    CP     (IY+04)       m. Exp2 vergleichen
35A6  38 3A       JR     C,35E2        Exp1<Exp2? d. entspr. behand.
35A8  20 33       JR     NZ,35DD       Exp1>Exp2? d. entspr. behand.
35AA  B7          OR     A             beide Exp =0?
35AB  C8          RET    Z             dann Arg1=Arg2=0, zurück
35AC  DD 7E 03    LD     A,(IX+03)     sonst Vorzeichen
35AF  FD AE 03    XOR    (IY+03)       vergleichen
35B2  FA DD 35    JP     M,35DD        versch. Vorz.? d. entspr. beh.
35B5  DD 7E 03    LD     A,(IX+03)       sonst
35B8  FD 96 03    SUB    (IY+03)         (Exp gleich, Vorz, gleich)
35BB  20 17       JR     NZ,35D4         Mantissen von oben
```

```
35BD  DD 7E 02     LD     A,(IX+02)       byteweise vergleichen
35C0  FD 96 02     SUB    (IY+02)
35C3  20 0F        JR     NZ,35D4         bei Ungleichheit
35C5  DD 7E 01     LD     A,(IX+01)       raus
35C8  FD 96 01     SUB    (IY+01)
35CB  20 07        JR     NZ,35D4
35CD  DD 7E 00     LD     A,(IX+00)
35D0  FD 96 00     SUB    (IY+00)
35D3  C8           RET    Z
35D4  9F           SBC    A             A:=$FF, wenn Arg1<Arg2
35D5  FD AE 03     XOR    (IY+03)       bei negativen Vorz. b7 invert.
35D8  87           ADD    A             und ins Carry
35D9  9F           SBC    A             von dort in ganzen Akku
35DA  D8           RET    C             wenn Arg1<Arg2, raus
35DB  3C           INC    A             A:=01, wenn Arg1>Arg2
35DC  C9           RET

35DD  DD 7E 03     LD     A,(IX+03)     Vorz. Arg1 als Vergleichserg.
35E0  18 F6        JR     35D8          setzen

35E2  FD 7E 03     LD     A,(IY+03)     Vorz. Arg2
35E5  2F           CPL                  invertiert als Vergleichserg.
35E6  18 F0        JR     35D8          setzen

***********************************   FLO SGN-Funktion
                                      IN : (HL): FLO Zahl
                                      OUT: A:= SGN(Arg)
                                             A=$00,Z=1,CY=0: Arg=0
                                             A=$01,S=0,CY=0: Arg>0
                                             A=$FF,S=1,CY=1: Arg<0
35E8  E5           PUSH   HL            Zeiger auf Arg
35E9  DD E1        POP    IX            nach IX
35EB  DD 7E 04     LD     A,(IX+04)     Exp d. Arg
35EE  B7           OR     A             Arg =0?
35EF  C8           RET    Z             d. A:=0, zurück
35F0  DD 7E 03     LD     A,(IX+03)     sonst Vorz. d. Arg
35F3  87           ADD    A             ins Carry
35F4  9F           SBC    A             von dort in den ganzen Akku
35F5  D8           RET    C             Arg<0? dann zurück
35F6  3C           INC    A             Arg>0? d. A:=01
35F7  C9           RET

***********************************   Vorzeichen invertieren
                                      IN : (HL): FLO Zahl, Arg
                                      OUT: (HL): -Arg
35F8  E5           PUSH   HL            Zeiger auf Arg
35F9  DD E1        POP    IX            nach IX
35FB  DD 7E 03     LD     A,(IX+03)     Vorzeichen
35FE  EE 80        XOR    80            invertieren
3600  DD 77 03     LD     (IX+03),A     und neu abspeichern
3603  C9           RET
```

```
***********************************   FLO FIX-Funktion
                                      IN : (IX)=(HL): FLO Zahl, Arg
                                           A: Exp
                                      OUT: (HL): FIX(Arg)
                                           B: Vorzeichen; C: Mantissenlänge
                                           A,E: Rundungsbyte
3604  AF           XOR   A              2er Komplement
3605  DD 96 04     SUB   (IX+04)        d. Exp bilden
3608  20 0A        JR    NZ,3614        <>0? d. Nachkommastellen elim.
360A  06 04        LD    B,04           sonst alle 4 Mantissenbytes
360C  77           LD    (HL),A         auf Null setzen
360D  23           INC   HL             Zeiger auf nächstes Byte
360E  10 FC        DJNZ  360C
3610  0E 01        LD    C,01           Mantissenlänge auf 1 Byte
3612  37           SCF                  CY:=1 f. o.k.
3613  C9           RET

***********************************   Nachkommastellen abschneiden
                                      IN : (IX)=(HL): FLO Zahl
                                           A: Exp, 2er Komplem.
                                      OUT: (HL): 4-Byte Integer
                                           A,E: Rundungsbyte
                                           S: Rundungsbit
                                           B: Vorzeichen
                                           C: Mantissenlänge
                                           CY=1, wenn o.k.
3614  C6 A0        ADD   A0             32-Exp=Anz. zu shiftend. Bits
3616  D0           RET   NC             Exp>$A0? dann Überlauf, raus
3617  E5           PUSH  HL             Zeiger auf Arg retten
3618  CD 3D 36     CALL  363D           FAC nach rechts shiften
361B  AF           XOR   A
361C  B8           CP    B              Nachkommaflag
361D  8F           ADC   A              A:=01, wenn Nachkommastellen
361E  B1           OR    C              und Rundungsbyte
361F  4D           LD    C,L              unteres Word
3620  44           LD    B,H              nach BC retten
3621  E1           POP   HL             Zeiger auf Zahl
3622  71           LD    (HL),C           untere 3 Bytes
3623  23           INC   HL               wieder
3624  70           LD    (HL),B           als Integer speichern
3625  23           INC   HL
3626  73           LD    (HL),E
3627  23           INC   HL
3628  5F           LD    E,A            Rundungsbyte nach E retten
3629  7E           LD    A,(HL)         Vorzeichen der Zahl nach A
362A  72           LD    (HL),D         1.MSB d. entstandenen Integers
362B  E6 80        AND   80             Vorzeichen isolieren
362D  47           LD    B,A            und nach B
362E  0E 04        LD    C,04           Mantissenlänge setzen
3630  AF           XOR   A
3631  B6           OR    (HL)           oberes Mantissenbyte
3632  20 05        JR    NZ,3639        <>0? dann aus Schleife raus
3634  2B           DEC   HL             Zeiger auf nächst nieder. Byte
3635  0D           DEC   C              Länge des Integers erniedrigen
3636  20 F9        JR    NZ,3631        noch nicht null? dann weiter
3638  0C           INC   C              sonst Mantissenlänge:=1
3639  7B           LD    A,E            Rundungsbyte
363A  B7           OR    A              testen: Rundungsbit b7 -> S
```

```
363B  37        SCF               CY:=1 f. o.k.
363C  C9        RET

*********************************  FLO Arg rechtsverschieben
                                   IN : (HL): 4-Byte Mantisse
                                          A: Anz. zu shiftender Bits
                                   OUT: DE,HL: geshiftete Mantisse
                                          C: Rundungsbyte
                                          B: Nachkommaflag (=0:Integ.)
363D  FE 21     CP     21            weniger als 33 Bits shiften?
363F  38 02     JR     C,3643        dann o.k.
3641  3E 21     LD     A,21          sonst Zahl auf 33 Bits setzen
3643  5E        LD     E,(HL)          Mantisse
3644  23        INC    HL              von (HL)
3645  56        LD     D,(HL)          nach DE,HL
3646  23        INC    HL              übernehmen
3647  4E        LD     C,(HL)
3648  23        INC    HL
3649  66        LD     H,(HL)
364A  69        LD     L,C
364B  EB        EX     DE,HL
364C  CB FA     SET    7,D           Vorz. durch 1 ersetzen
364E  01 00 00  LD     BC,0000       Nachkommaflag u. Rundungsbyte
3651  18 0B     JR     365E          Schleife überspringen
3653  4F        LD     C,A           Bitzähler retten
3654  78        LD     A,B           Nachkommaflag
3655  B5        OR     L             und unterstes Byte verknüpfen
3656  47        LD     B,A           und als neues Nachkommaflag
3657  79        LD     A,C           Bitzähler
3658  4D        LD     C,L             gesamte Mantisse
3659  6C        LD     L,H             8 Bits = 1 Byte
365A  63        LD     H,E             rechtsverschieben
365B  5A        LD     E,D
365C  16 00     LD     D,00          Null ins 1.MSB
365E  D6 08     SUB    08            Bitzähler f. 1 Byte rechtsver.
3660  30 F1     JR     NC,3653       kein Borger? d. noch ein Byte
3662  C6 08     ADD    08            Fehler ausgleichen
3664  C8        RET    Z             ging auf? dann o.k., raus
3665  CB 3A     SRL    D               gesamte Mantisse
3667  CB 1B     RR     E               um 1 Bit
3669  CB 1C     RR     H               rechtsverschieben
366B  CB 1D     RR     L
366D  CB 19     RR     C
366F  3D        DEC    A             Bitzähler entspr. erniedrigen
3670  20 F3     JR     NZ,3665       noch nicht fertig? d. weiter
3672  C9        RET

*********************************  FLO Zahl normalisieren
                                   IN : DE,HL: 4-Byte Mantisse
                                          A: Exponent; C: Rundungsbyte
                                   OUT: DE,HL: normalis. Mantisse
                                          A: neuer Exponent
3673  14        INC    D             oberstes Mantissenbyte
3674  15        DEC    D             testen
3675  F8        RET    M             oberstes Bit gesetzt? d. raus
3676  20 17     JR     NZ,368F       1.MSB<>0? d. bitweise normal.
3678  57        LD     D,A           Exp retten
3679  7B        LD     A,E             restliche
```

```
367A  B4        OR    H             Mantisse
367B  B5        OR    L             (incl. Rundungsbyte)
367C  B1        OR    C             =0 ?
367D  C8        RET   Z           dann raus, Zahl:=0
367E  7A        LD    A,D         sonst Exp zurück
367F  D6 08     SUB   08          -8, f. 1 Byte linksverschieben
3681  38 1C     JR    C,369F      Unterlauf? dann raus, CY:=0
3683  C8        RET   Z           Null? dann Zahl:=0, raus
3684  53        LD    D,E           sonst gesamte
3685  5C        LD    E,H           Mantisse
3686  65        LD    H,L           (incl. Rundungsbyte)
3687  69        LD    L,C           8 Bits = 1Byte
3688  0E 00     LD    C,00          linksverschieben
368A  14        INC   D           oberstes Byte
368B  15        DEC   D           testen
368C  28 F1     JR    Z,367F      =0? d. weitere 8 Bits versch.
368E  F8        RET   M           b7 gesetzt? dann o.k., raus
368F  3D        DEC   A           sonst f. 1 Bit linksversch.
3690  C8        RET   Z           Exp =0? dann Zahl:=0, raus
3691  CB 21     SLA   C             sonst gesamte
3693  CB 15     RL    L             Mantisse
3695  CB 14     RL    H             (incl. Rundungsbyte)
3697  CB 13     RL    E             um 1 Bit
3699  CB 12     RL    D             linksverschieben
369B  F2 8F 36  JP    P,368F      bis oberstes Bit gesetzt
369E  C9        RET               danach raus
369F  AF        XOR   A           CY:=0 f. Fehler
36A0  C9        RET

***********************************  FLO 4-Byte Mantisse>Real
                                     IN : (HL): 4-Byte Integer
                                            B: Exponent
                                            A: Vorzeichen
                                            C: Rundungsbyte
36A1  E5        PUSH  HL          Zeiger auf Mantisse
36A2  DD E1     POP   IX          nach IX
36A4  DD 70 04  LD    (IX+04),B   Exp ablegen
36A7  47        LD    B,A         Vorzeichen nach B retten
36A8  5E        LD    E,(HL)        Mantisse
36A9  23        INC   HL            nach
36AA  56        LD    D,(HL)        DE,HL
36AB  23        INC   HL            holen
36AC  7E        LD    A,(HL)
36AD  23        INC   HL
36AE  66        LD    H,(HL)
36AF  6F        LD    L,A
36B0  EB        EX    DE,HL
36B1  DD 7E 04  LD    A,(IX+04)   Exp wieder zurück
36B4  CD 73 36  CALL  3673        Zahl normalisieren
36B7  DD 77 04  LD    (IX+04),A   und neuen Exp speichern

***********************************  Mantisse runden, Vorz. eintragen
                                     IN : (IX): Rückgabeadr. f. Mant.
                                            DE,HL: Mantisse, 4 Bytes
                                            B<b7>: Vorzeichen
                                            (IX+04): Exponent
                                     OUT: (HL): FLO Zahl
36BA  CB 21     SLA   C           Rundungsbit ins Carry
```

```
36BC  30 13       JR    NC,36D1      =0? dann nicht aufrunden
36BE  2C          INC   L              sonst
36BF  20 10       JR    NZ,36D1        Mantisse um
36C1  24          INC   H              1 erhöhen
36C2  20 0D       JR    NZ,36D1
36C4  1C          INC   E
36C5  20 0A       JR    NZ,36D1
36C7  14          INC   D
36C8  20 07       JR    NZ,36D1      Übertrag auf Exp?
36CA  DD 34 04    INC   (IX+04)      d. Exp erhöhen
36CD  28 1F       JR    Z,36EE       Übertrag? dann Überlauf
36CF  16 80       LD    D,80         sonst D setzen
36D1  78          LD    A,B          Vorzeichen
36D2  F6 7F       OR    7F           b7 isolieren
36D4  A2          AND   D            Vorzeichen ins 1.MSB hinein
36D5  DD 77 03    LD    (IX+03),A    und Ergebnis als 1.MSB setzen
36D8  DD 73 02    LD    (IX+02),E      restl.
36DB  DD 74 01    LD    (IX+01),H      Mantisse
36DE  DD 75 00    LD    (IX+00),L      speichern
36E1  DD E5       PUSH  IX           Zeiger auf Zahl
36E3  E1          POP   HL           nach HL
36E4  37          SCF                CY:=1 f. o.k.
36E5  C9          RET

**********************************  Unterlauf, Null setzen
                                    IN : (IX): FLO Zahl
                                    OUT: FLO Null bei IX u. HL
36E6  AF          XOR   A              Null
36E7  DD 77 04    LD    (IX+04),A      als Exp setzen
36EA  18 F5       JR    36E1           Zeiger nach HL, raus

**********************************  Überlauf, max. pos. Zahl setzen
                                    IN : (IX): FLO Zahl
                                    OUT: (IX),(HL): max. pos. Zahl
36EC  06 00       LD    B,00           positives Vorzeichen setzen

**********************************  Überlauf, max. Betrag setzen
                                    IN : (IX): FLO Zahl
                                           B<b7>: Vorzeichen d. Zahl
                                    OUT: (IX): max. FLO Betrag
36EE  78          LD    A,B            Vorzeichen
36EF  F6 7F       OR    7F             nicht verändern
36F1  DD 77 03    LD    (IX+03),A      und speichern
36F4  F6 FF       OR    FF               sonst
36F6  DD 77 04    LD    (IX+04),A        gesamte Mantisse
36F9  DD 77 00    LD    (IX+00),A        und Exponenten
36FC  DD 77 01    LD    (IX+01),A        auf maximalen Wert
36FF  DD 77 02    LD    (IX+02),A        ($FF) setzen
3702  C9          RET

3703  C7          RST   00
3704  C7          RST   00
3705  C7          RST   00
3706  C7          RST   00
3707  C7          RST   00
```

```
-------------------- INTEGER ARITHMETICS (INT) --------------------

**********************************   INT Integer f. Dez.-Wand. aufber.
                                     IN : HL: 16-Bit 2er Kompl. Integ.
                                     OUT: HL: 15-Bit unsigned Integer
                                          B<b7>: Vorzeichen
                                          C:=2 als Mantissenlänge
                                          E:=0 f. Integer
3708  44          LD    B,H            Vorzeichen
3709  CD D1 37    CALL  37D1           Absolutwert bilden
370C  18 02       JR    3710           Kommapos. u. Mant.-Länge setz.

**********************************   INT pos. Integer, Dez.-Wand.-Par.
                                     IN : HL: unsigned Integer
                                     OUT: HL: unsigned Integer
                                          B<b7>:=0 f. positiv
                                          C:=2 als Mantissenlänge
                                          E:=0 f. Integer
370E  06 00       LD    B,00           Vorzeichen positiv
3710  1E 00       LD    E,00           Kommaposition f. Integer
3712  0E 02       LD    C,02           Mantissenlänge
3714  C9          RET

**********************************   INT signed Binary>2er Kompl.
                                     IN : HL: unsigned Integer
                                          B<b7>: Vorzeichen
                                     OUT: HL: 2er Kompl. Integer
                                          CY=1, Z=0 f. Überlauf
3715  7C          LD    A,H            Hi-Byte d. Integer
3716  B7          OR    A              Integer >32767?
3717  FA 20 37    JP    M,3720         d. nur bei $8000 wandeln
371A  B0          OR    B              sonst Vorzeichen
371B  FA D4 37    JP    M,37D4         Vorzeichenwechsel? d. wandeln
371E  37          SCF                  sonst CY:=1 f. o.k.
371F  C9          RET

3720  EE 80       XOR   80             Integer
3722  B5          OR    L              <>$8000?
3723  C0          RET   NZ             dann CY:=0 f. Fehler, raus
3724  78          LD    A,B            sonst Vorzeichen
3725  37          SCF
3726  8F          ADC   A              ins Carry, o.k. b. negativ
3727  C9          RET                  Fehler bei positiv

**********************************   INT HL:=HL+DE
                                     IN : HL: Integerarg1, 2er Kompl.
                                          DE: Integerarg2, 2er Kompl.
                                     OUT: HL:=HL+DE
3728  B7          OR    A              CY:=0 um ADC f. Parity ermögl.
3729  ED 5A       ADC   HL,DE          2er Kompl. Integer addieren
372B  37          SCF                  CY:=1 f. o.k.
372C  E0          RET   PO             kein Überlauf? dann o.k., raus
372D  F6 FF       OR    FF             sonst Z:=0, CY:=0 f. Fehler
372F  C9          RET
```

```
***********************************   INT HL:=DE-HL
                                      IN : HL: Integerarg1, 2er Kompl.
                                           DE: Integerarg2, 2er Kompl.
                                      OUT: HL:=DE-HL
3730  EB           EX     DE,HL

***********************************   INT HL:=HL-DE
                                      IN : HL: Integerarg1, 2er Kompl.
                                           DE: Integerarg2, 2er Kompl.
                                      OUT: HL:=HL-DE
3731  B7           OR     A             CY:=0 f. SBC
3732  ED 52        SBC    HL,DE         Argumente subtrahieren
3734  37           SCF                  CY:=1 f. o.k.
3735  E0           RET    PO            kein Überlauf? dann o.k., raus
3736  F6 FF        OR     FF            sonst Z:=0, CY:=0 f. Fehler
3738  C9           RET

***********************************   INT HL:=HL*DE
                                      IN : HL: Integerarg1, 2er Kompl.
                                           DE: Integerarg2, 2er Kompl.
                                      OUT: HL:=ABS(HL)*ABS(DE)
                                           B:Vorzeichen d. Ergebnisses
3739  CD 45 37     CALL   3745          Ergebnisvorz. u. Absolutwerte
373C  CD 50 37     CALL   3750          vorzeichenlose Multiplikation
373F  D2 15 37     JP     NC,3715       kein Überl.? d. in 2er Kompl.
3742  F6 FF        OR     FF            sonst Z:=0, CY:=0 f. Fehler
3744  C9           RET

***********************************   Vorz. d. Ergebn. bestimmen
                                      IN : HL: Integer, 2er Kompl.
                                           DE: Integer, 2er Kompl.
                                      OUT: HL:=ABS(HL)
                                           DE:=ABS(DE)
                                           B: Vorzeichenvergleich
3745  7C           LD     A,H           Vorz. Int1
3746  AA           XOR    D             mit Vorz. Int2 vergleichen
3747  47           LD     B,A           und Verleichsergbenis nach B
3748  EB           EX     DE,HL         Integer 2
3749  CD D1 37     CALL   37D1          Absolutwert bilden
374C  EB           EX     DE,HL         Integer 1
374D  C3 D1 37     JP     37D1          Absolutwert bilden, raus

***********************************   vorzeichenlose Multiplikation
                                      IN : HL: unsigned Integer, Int1
                                           DE: unsigned Integer, Int2
                                      OUT: HL: unsigned Integer, Erg.
                                           CY=1, wenn Fehler
3750  7C           LD     A,H           Hi-Byte Int1
3751  B7           OR     A             Null?
3752  28 05        JR     Z,3759        dann o.k.
3754  7A           LD     A,D           Hi-Byte Int2
3755  B7           OR     A             gleich Null?
3756  37           SCF                  CY:=1 f. Fehler, wenn
3757  C0           RET    NZ            beide Hi-Bytes<>0? dann Fehler
3758  EB           EX     DE,HL         Byteoperand nach HL
3759  B5           OR     L             Lo-Byte auch Null?
375A  C8           RET    Z             dann Erg:=0, raus
375B  7A           LD     A,D           Hi-Byte Wordoperand
```

```
375C  B3          OR     E          Lo-Byte Wordoperand
375D  7D          LD     A,L        Byteoperand retten
375E  6B          LD     L,E          Wordoperand
375F  62          LD     H,D          nach HL
3760  C8          RET    Z          =0? d. Erg.:=0, raus
3761  FE 03       CP     03         Zahl1
3763  38 10       JR     C,3775     <3? dann speziell behandeln
3765  37          SCF               CY:=1 als Byteende-Markierung
3766  8F          ADC    A          Byteoperanden bitweise in CY
3767  30 FD       JR     NC,3766    bis zum ersten 1-Bit
3769  29          ADD    HL,HL      lfd. Ergebnis verdoppeln
376A  D8          RET    C          Überlauf? dann raus
376B  87          ADD    A          lfd. Bit d. Bitoperanden
376C  30 02       JR     NC,3770    nicht gesetzt? dann weiter
376E  19          ADD    HL,DE      sonst Byteoperand addieren
376F  D8          RET    C          Überlauf? dann Fehler, raus
3770  FE 80       CP     80         Byteende erreicht?
3772  20 F5       JR     NZ,3769    sonst nächstes Bit bearbeiten
3774  C9          RET

3775  FE 01       CP     01         Byteoperand=1?
3777  C8          RET    Z          d. Erg:= Wordoperand
3778  29          ADD    HL,HL      sonst Erg.:=Wordoperand*2
3779  C9          RET

***********************************  INT HL:=HL DIV DE
                                     IN : HL: Integer1, 2er Kompl.
                                          DE: Integer2, 2er Kompl.
                                     OUT: HL:=HL DIV DE, 2er Kompl.
                                          CY=0 f. Fehler
                                            Z=1: 'division by zero'
                                            Z=0: 'overflow'
377A  CD 89 37    CALL   3789          Division
377D  DA 15 37    JP     C,3715        o.k.? d. Vorz. in B übernehm.
3780  C9          RET

***********************************  INT HL:=HL MOD DE
                                     IN : HL: Integer1, 2er Kompl.
                                          DE: Integer2, 2er Kompl.
                                     OUT: HL:=HL MOD DE, 2er Kompl.
3781  4C          LD     C,H           Vorz. d. Dividenden retten
3782  CD 89 37    CALL   3789          Division
3785  EB          EX     DE,HL         Rest der Division nach HL
3786  41          LD     B,C           Vorzeichen d. Dividenden
3787  18 F4       JR     377D          in Rest übernehmen, wenn o.k.

***********************************  INT Division
                                     IN : HL: Integer1, 2er Kompl.
                                          DE: Integer2, 2er kompl.
                                     OUT: HL:=ABS(HL DIV DE)
                                          DE:=ABS(HL MOD DE)
                                          B: Vorzeichen d. Quotienten
                                          Z=1, f. "division by zero"
3789  CD 45 37    CALL   3745          Vorz. d. Ergebn. nach B
```

```
***********************************   INT vorzeichenlose Division
                                      IN : HL: Integer1, unsigned
                                           DE: Integer2, unsigned
                                      OUT: HL:=HL DIV DE
                                           DE:=HL MOD DE
                                           Z=1, f. "division by zero"
378C  7A          LD    A,D              Divisor
378D  B3          OR    E                =0?
378E  C8          RET   Z              d. Flag f. "division by zero"
378F  C5          PUSH  BC             Vorz. d. Ergebn. retten
3790  EB          EX    DE,HL          Argumente vertauschen
3791  06 01       LD    B,01           Bitzähler
3793  7C          LD    A,H            Hi-Byte Int2
3794  B7          OR    A              <>0?
3795  20 09       JR    NZ,37A0        d. bitweise shiften
3797  7A          LD    A,D            Hi-Byte d. Int1
3798  BD          CP    L              Lo-Byte Int2 davon abziehbar
3799  38 05       JR    C,37A0         sonst bitweise shiften
379B  65          LD    H,L              sonst Divisor bereits um
379C  2E 00       LD    L,00             1 Byte linksverschieben
379E  06 09       LD    B,09           und Bitzähler entspr. setzen
37A0  7B          LD    A,E              Divisor
37A1  95          SUB   L                vom Dividend abziehbar?
37A2  7A          LD    A,D
37A3  9C          SBC   H
37A4  38 05       JR    C,37AB         sonst raus aus der Schleife
37A6  04          INC   B              Bitzähler erhöhen
37A7  29          ADD   HL,HL          Divisor versch., b. Int1<Int2
37A8  30 F6       JR    NC,37A0        und, wenn o.k., nochmals prüf.
37AA  3F          CCF
37AB  3F          CCF
37AC  78          LD    A,B            Zähler f. Linksverschiebungen
37AD  44          LD    B,H            neuer Divisor nach BC
37AE  4D          LD    C,L
37AF  21 00 00    LD    HL,0000        lfd. Ergebnis zurücksetzen
37B2  3D          DEC   A              Bitzähler
37B3  20 03       JR    NZ,37B8        <>0? dann normal anfangen
37B5  18 17       JR    37CE           sonst Dividend<Divisor, raus
37B7  29          ADD   HL,HL          Ergebnis mit 2 multiplizier.
37B8  F5          PUSH  AF             Zähler retten
37B9  78          LD    A,B              Divisor
37BA  1F          RRA                    um 1 Bit
37BB  47          LD    B,A              nach rechts
37BC  79          LD    A,C              verschieben
37BD  1F          RRA
37BE  4F          LD    C,A
37BF  7B          LD    A,E              und mit Rest
37C0  91          SUB   C                vergleichen
37C1  7A          LD    A,D
37C2  98          SBC   B
37C3  38 05       JR    C,37CA         Divisor kleiner? dann weiter
37C5  57          LD    D,A              sonst
37C6  7B          LD    A,E              Rest:=Rest-Divisor
37C7  91          SUB   C
37C8  5F          LD    E,A
37C9  2C          INC   L              und b0 d. lfd. Erg. setzen
37CA  F1          POP   AF             Bitzähler
37CB  3D          DEC   A              herunterzählen
```

```
37CC  20 E9      JR     NZ,37B7      <>0? dann nochmal durchlaufen
37CE  37         SCF                 sonst CY:=1 f. o.k.
37CF  C1         POP    BC           Vorz. d. Ergebnisses
37D0  C9         RET

*********************************** INT HL:=ABS(HL)
                                   IN : HL: Integer, 2er Kompl.
                                   OUT: HL:=ABS(HL), 2er Kompl.
37D1  7C         LD     A,H            Vorzeichen d. Integers
37D2  B7         OR     A              testen
37D3  F0         RET    P              zurück, wenn Zahl positiv

*********************************** INT HL:=-HL
                                   IN : HL: Integer, 2er Kompl.
                                   OUT: HL:=-HL, 2er Kompl.
37D4  AF         XOR    A                Zweierkomplement
37D5  95         SUB    L                der beiden
37D6  6F         LD     L,A              Bytes des Integers
37D7  9C         SBC    H                bilden
37D8  95         SUB    L
37D9  BC         CP     H              (neues Hi-Byte=altes Hi-Byte?)
37DA  67         LD     H,A
37DB  37         SCF
37DC  C0         RET    NZ             wenn Hi-Byte verändert, zurück
37DD  FE 01      CP     01             sonst CY:=0 f. Überl. b. $8000
37DF  C9         RET

*********************************** INT A:=SGN(HL)
                                   IN : HL: Integer,2er Kompl.
                                   OUT: A:=SGN(HL)
37E0  7C         LD     A,H            Hi-Byte
37E1  87         ADD    A              Vorzeichen ins Carry
37E2  9F         SBC    A              und von dort in ganzen Akku
37E3  D8         RET    C              Integer negativ? d. A:=$FF
37E4  B5         OR     L              Integer null?
37E5  C8         RET    Z              dann A:=0, raus
37E6  AF         XOR    A              sonst
37E7  3C         INC    A              A:=01 f. Integer positiv
37E8  C9         RET

*********************************** INT Vergleich HL-DE
                                   IN : HL: Integer1, 2er Kompl.
                                        DE: Integer2, 2er Kompl.
                                   OUT: A:=$01, Int1>Int2
                                        A:=$00, Int1=Int2
                                        A:=$FF, Int1<Int2
                                          (A:=SGN(HL-DE))
37E9  7C         LD     A,H            Vorzeichen
37EA  AA         XOR    D              vergleichen
37EB  7C         LD     A,H            Vorzeichen Int1
37EC  F2 F4 37   JP     P,37F4         Vorz. gleich? d. entspr. ausw.
37EF  87         ADD    A              sonst (Vorzeichen ungl.)
37F0  9F         SBC    A              Vorz. Int1 in ganz A und CY
37F1  D8         RET    C              CY=1 bzw. A=$FF? dann raus
37F2  3C         INC    A              sonst A:=1 setzen
37F3  C9         RET
37F4  BA         CP     D              Hi-Bytes vergleichen
37F5  20 F9      JR     NZ,37F0        ungleich? dann raus, CY->A
```

```
37F7  7D          LD     A,L           Lo-Bytes
37F8  93          SUB    E             vergleichen
37F9  20 F5       JR     NZ,37F0       ungleich? dann raus, CY->A
37FB  C9          RET                  sonst gleich, A:=00

37FC  C7          RST    00
37FD  C7          RST    00
37FE  C7          RST    00
37FF  01


------------------------ Der Zeichensatz -------------------------


-----------------------------------    CHR$(0)
3800  FF                                  ''''''''
3801  C3                                  ''    ''
3802  C3                                  ''    ''
3803  C3                                  ''    ''
3804  C3                                  ''    ''
3805  C3                                  ''    ''
3806  C3                                  ''    ''
3807  FF                                  ''''''''

-----------------------------------    CHR$(1)
3808  FF                                  ''''''''
3809  C0                                  ''
380A  C0                                  ''
380B  C0                                  ''
380C  C0                                  ''
380D  C0                                  ''
380E  C0                                  ''
380F  C0                                  ''

-----------------------------------    CHR$(2)
3810  18                                     ''
3811  18                                     ''
3812  18                                     ''
3813  18                                     ''
3814  18                                     ''
3815  18                                     ''
3816  18                                     ''
3817  FF                                  ''''''''

-----------------------------------    CHR$(3)
3818  03                                        ''
3819  03                                        ''
381A  03                                        ''
381B  03                                        ''
381C  03                                        ''
381D  03                                        ''
381E  03                                        ''
381F  FF                                  ''''''''
```

```
----------------------------------        CHR$(4)
3820   0C                                     II
3821   18                                    II
3822   30                                   II
3823   7E                                  IIIIII
3824   0C                                     II
3825   18                                    II
3826   30                                   II
3827   00

----------------------------------        CHR$(5)
3828   FF                                 IIIIIIII
3829   C3                                 II    II
382A   E7                                 III  III
382B   DB                                 II II II
382C   DB                                 II II II
382D   E7                                 III  III
382E   C3                                 II    II
382F   FF                                 IIIIIIII

----------------------------------        CHR$(6)
3830   00
3831   01                                        I
3832   03                                       II
3833   06                                      II
3834   CC                                 II  II
3835   78                                  IIII
3836   30                                   II
3837   00

----------------------------------        CHR$(7)
3838   3C                                   IIII
3839   66                                  II  II
383A   C3                                 II    II
383B   C3                                 II    II
383C   FF                                 IIIIIIII
383D   24                                   I  I
383E   E7                                 III  III
383F   00

----------------------------------        CHR$(8)
3840   00
3841   00
3842   30                                   II
3843   60                                  II
3844   FF                                 IIIIIIII
3845   60                                  II
3846   30                                   II
3847   00
```

```
------------------------------------  CHR$(9)
3848   00
3849   00
384A   0C                                 ''
384B   06                                  ''
384C   FF                             ''''''''
384D   06                                  ''
384E   0C                                 ''
384F   00

------------------------------------  CHR$(10)
3850   18                                ''
3851   18                                ''
3852   18                                ''
3853   18                                ''
3854   DB                             '' '' ''
3855   7E                              ''''''
3856   3C                               ''''
3857   18                                ''

------------------------------------  CHR$(11)
3858   18                                ''
3859   3C                               ''''
385A   7E                              ''''''
385B   DB                             '' '' ''
385C   18                                ''
385D   18                                ''
385E   18                                ''
385F   18                                ''

------------------------------------  CHR$(12)
3860   18                                ''
3861   5A                              ' '' '
3862   3C                               ''''
3863   99                             '  ''  '
3864   DB                             '' '' ''
3865   7E                              ''''''
3866   3C                               ''''
3867   18                                ''

------------------------------------  CHR$(13)
3868   00
3869   03                                   ''
386A   33                               ''  ''
386B   63                              ''   ''
386C   FE                             '''''''
386D   60                              ''
386E   30                               ''
386F   00
```

```
-----------------------------------   CHR$(14)
3870  3C                                ''''
3871  66                               ''  ''
3872  FF                              ''''''''
3873  DB                              '' '' ''
3874  DB                              '' '' ''
3875  FF                              ''''''''
3876  66                               ''  ''
3877  3C                                ''''

-----------------------------------   CHR$(15)
3878  3C                                ''''
3879  66                               ''  ''
387A  C3                              ''    ''
387B  DB                              '' '' ''
387C  DB                              '' '' ''
387D  C3                              ''    ''
387E  66                               ''  ''
387F  3C                                ''''

-----------------------------------   CHR$(16)
3880  FF                              ''''''''
3881  C3                              ''    ''
3882  C3                              ''    ''
3883  FF                              ''''''''
3884  C3                              ''    ''
3885  C3                              ''    ''
3886  C3                              ''    ''
3887  FF                              ''''''''

-----------------------------------   CHR$(17)
3888  3C                                ''''
3889  7E                               ''''''
388A  DB                              '' '' ''
388B  DB                              '' '' ''
388C  DF                              '' '''''
388D  C3                              ''    ''
388E  66                               ''  ''
388F  3C                                ''''

-----------------------------------   CHR$(18)
3890  3C                                ''''
3891  66                               ''  ''
3892  C3                              ''    ''
3893  DF                              '' '''''
3894  DB                              '' '' ''
3895  DB                              '' '' ''
3896  7E                               ''''''
3897  3C                                ''''
```

```
----------------------------------  CHR$(19)
3898  3C                                 ''''
3899  66                                ''  ''
389A  C3                               ''    ''
389B  FB                               ''''' ''
389C  DB                               '' '' ''
389D  DB                               '' '' ''
389E  7E                                ''''''
389F  3C                                 ''''

----------------------------------  CHR$(20)
38A0  3C                                 ''''
38A1  7E                                ''''''
38A2  DB                               '' '' ''
38A3  DB                               '' '' ''
38A4  FB                               ''''' ''
38A5  C3                               ''    ''
38A6  66                                ''  ''
38A7  3C                                 ''''

----------------------------------  CHR$(21)
38A8  00
38A9  01                                      '
38AA  33                                 ''  ''
38AB  1E                                  ''''
38AC  CE                               ''  '''
38AD  7B                                '''' ''
38AE  31                                 ''   '
38AF  00

----------------------------------  CHR$(22)
38B0  7E                                ''''''
38B1  66                                ''  ''
38B2  66                                ''  ''
38B3  66                                ''  ''
38B4  66                                ''  ''
38B5  66                                ''  ''
38B6  66                                ''  ''
38B7  E7                               '''  '''

----------------------------------  CHR$(23)
38B8  03                                     ''
38B9  03                                     ''
38BA  03                                     ''
38BB  FF                               ''''''''
38BC  03                                     ''
38BD  03                                     ''
38BE  03                                     ''
38BF  00
```

```
-----------------------------------     CHR$(24)
38C0  FF                                   ''''''''
38C1  66                                    ''  '' 
38C2  3C                                     ''''  
38C3  18                                      ''   
38C4  18                                      ''   
38C5  3C                                     ''''  
38C6  66                                    ''  '' 
38C7  FF                                   ''''''''

-----------------------------------     CHR$(25)
38C8  18                                      ''   
38C9  18                                      ''   
38CA  3C                                     ''''  
38CB  3C                                     ''''  
38CC  3C                                     ''''  
38CD  3C                                     ''''  
38CE  18                                      ''   
38CF  18                                      ''   

-----------------------------------     CHR$(26)
38D0  3C                                     ''''  
38D1  66                                    ''  '' 
38D2  66                                    ''  '' 
38D3  30                                     ''    
38D4  18                                      ''   
38D5  00
38D6  18                                      ''   
38D7  00

-----------------------------------     CHR$(27)
38D8  3C                                     ''''  
38D9  66                                    ''  '' 
38DA  C3                                   ''    ''
38DB  FF                                   ''''''''
38DC  C3                                   ''    ''
38DD  C3                                   ''    ''
38DE  66                                    ''  '' 
38DF  3C                                     ''''  

-----------------------------------     CHR$(28)
38E0  FF                                   ''''''''
38E1  DB                                   '' '' ''
38E2  DB                                   '' '' ''
38E3  DB                                   '' '' ''
38E4  FB                                   ''''' ''
38E5  C3                                   ''    ''
38E6  C3                                   ''    ''
38E7  FF                                   ''''''''
```

```
----------------------------------  CHR$(29)
38E8  FF                               ''''''''
38E9  C3                               ''    ''
38EA  C3                               ''    ''
38EB  FB                               ''''' ''
38EC  DB                               '' '' ''
38ED  DB                               '' '' ''
38EE  DB                               '' '' ''
38EF  FF                               ''''''''

----------------------------------  CHR$(30)
38F0  FF                               ''''''''
38F1  C3                               ''    ''
38F2  C3                               ''    ''
38F3  DF                               '' '''''
38F4  DB                               '' '' ''
38F5  DB                               '' '' ''
38F6  DB                               '' '' ''
38F7  FF                               ''''''''

----------------------------------  CHR$(31)
38F8  FF                               ''''''''
38F9  DB                               '' '' ''
38FA  DB                               '' '' ''
38FB  DB                               '' '' ''
38FC  DF                               '' '''''
38FD  C3                               ''    ''
38FE  C3                               ''    ''
38FF  FF                               ''''''''
3900  00 00 00 00 00 00 00 00
3908  18 18 18 18 18 00 18 00
3910  6C 6C 6C 00 00 00 00 00
3918  6C 6C FE 6C FE 6C 6C 00
3920  18 3E 58 3C 1A 7C 18 00
3928  00 C6 CC 18 30 66 C6 00
3930  38 6C 38 76 DC CC 76 00
3938  18 18 30 00 00 00 00 00
3940  0C 18 30 30 30 18 0C 00
3948  30 18 0C 0C 0C 18 30 00
3950  00 66 3C FF 3C 66 00 00
3958  00 18 18 7E 18 18 00 00
3960  00 00 00 00 00 18 18 30
3968  00 00 00 7E 00 00 00 00
3970  00 00 00 00 00 18 18 00
3978  06 0C 18 30 60 C0 80 00
3980  7C C6 CE D6 E6 C6 7C 00
3988  18 38 18 18 18 18 7E 00
3990  3C 66 06 3C 60 66 7E 00
3998  3C 66 06 1C 06 66 3C 00
39A0  1C 3C 6C CC FE 0C 1E 00
39A8  7E 62 60 7C 06 66 3C 00
39B0  3C 66 60 7C 66 66 3C 00
39B8  7E 66 06 0C 18 18 18 00
39C0  3C 66 66 3C 66 66 3C 00
39C8  3C 66 66 3E 06 66 3C 00
39D0  00 00 18 18 00 18 18 00
39D8  00 00 18 18 00 18 18 30
39E0  0C 18 30 60 30 18 0C 00
```

```
39E8  00 00 7E 00 00 7E 00 00
39F0  60 30 18 0C 18 30 60 00
39F8  3C 66 66 0C 18 00 18 00
3A00  7C C6 DE DE DE C0 7C 00
3A08  18 3C 66 66 7E 66 66 00
3A10  FC 66 66 7C 66 66 FC 00
3A18  3C 66 C0 C0 C0 66 3C 00
3A20  F8 6C 66 66 66 6C F8 00
3A28  FE 62 68 78 68 62 FE 00
3A30  FE 62 68 78 68 60 F0 00
3A38  3C 66 C0 C0 CE 66 3E 00
3A40  66 66 66 7E 66 66 66 00
3A48  7E 18 18 18 18 18 7E 00
3A50  1E 0C 0C 0C CC CC 78 00
3A58  E6 66 6C 78 6C 66 E6 00
3A60  F0 60 60 60 62 66 FE 00
3A68  C6 EE FE FE D6 C6 C6 00
3A70  C6 E6 F6 DE CE C6 C6 00
3A78  38 6C C6 C6 C6 6C 38 00
3A80  FC 66 66 7C 60 60 F0 00
3A88  38 6C C6 C6 DA CC 76 00
3A90  FC 66 66 7C 6C 66 E6 00
3A98  3C 66 60 3C 06 66 3C 00
3AA0  7E 5A 18 18 18 18 3C 00
3AA8  66 66 66 66 66 66 3C 00
3AB0  66 66 66 66 66 3C 18 00
3AB8  C6 C6 C6 D6 FE EE C6 00
3AC0  C6 6C 38 38 6C C6 C6 00
3AC8  66 66 66 3C 18 18 3C 00
3AD0  FE C6 8C 18 32 66 FE 00
3AD8  3C 30 30 30 30 30 3C 00
3AE0  C0 60 30 18 0C 06 02 00
3AE8  3C 0C 0C 0C 0C 0C 3C 00
3AF0  18 3C 7E 18 18 18 18 00
3AF8  00 00 00 00 00 00 00 FF
3B00  30 18 0C 00 00 00 00 00
3B08  00 00 78 0C 7C CC 76 00
3B10  E0 60 7C 66 66 66 DC 00
3B18  00 00 3C 66 60 66 3C 00
3B20  1C 0C 7C CC CC CC 76 00
3B28  00 00 3C 66 7E 60 3C 00
3B30  1C 36 30 78 30 30 78 00
3B38  00 00 3E 66 66 3E 06 7C
3B40  E0 60 6C 76 66 66 E6 00
3B48  18 00 38 18 18 18 3C 00
3B50  06 00 0E 06 06 66 66 3C
3B58  E0 60 66 6C 78 6C E6 00
3B60  38 18 18 18 18 18 3C 00
3B68  00 00 6C FE D6 D6 C6 00
3B70  00 00 DC 66 66 66 66 00
3B78  00 00 3C 66 66 66 3C 00
3B80  00 00 DC 66 66 7C 60 F0
3B88  00 00 76 CC CC 7C 0C 1E
3B90  00 00 DC 76 60 60 F0 00
3B98  00 00 3C 60 3C 06 7C 00
3BA0  30 30 7C 30 30 36 1C 00
3BA8  00 00 66 66 66 66 3E 00
3BB0  00 00 66 66 66 3C 18 00
```

```
3BB8  00 00 C6 D6 D6 FE 6C 00
3BC0  00 00 C6 6C 38 6C C6 00
3BC8  00 00 66 66 66 3E 06 7C
3BD0  00 00 7E 4C 18 32 7E 00
3BD8  0E 18 18 70 18 18 0E 00
3BE0  18 18 18 18 18 18 18 00
3BE8  70 18 18 0E 18 18 70 00
3BF0  76 DC 00 00 00 00 00 00
3BF8  CC 33 CC 33 CC 33 CC 33
3C00  00 00 00 00 00 00 00 00
3C08  F0 F0 F0 F0 00 00 00 00
3C10  0F 0F 0F 0F 00 00 00 00
3C18  FF FF FF FF 00 00 00 00
3C20  00 00 00 00 F0 F0 F0 F0
3C28  F0 F0 F0 F0 F0 F0 F0 F0
3C30  0F 0F 0F 0F F0 F0 F0 F0
3C38  FF FF FF FF F0 F0 F0 F0
3C40  00 00 00 00 0F 0F 0F 0F
3C48  F0 F0 F0 F0 0F 0F 0F 0F
3C50  0F 0F 0F 0F 0F 0F 0F 0F
3C58  FF FF FF FF 0F 0F 0F 0F
3C60  00 00 00 00 FF FF FF FF
3C68  F0 F0 F0 F0 FF FF FF FF
3C70  0F 0F 0F 0F FF FF FF FF
3C78  FF FF FF FF FF FF FF FF
3C80  00 00 00 18 18 00 00 00
3C88  18 18 18 18 18 00 00 00
3C90  00 00 00 1F 1F 00 00 00
3C98  18 18 18 1F 0F 00 00 00
3CA0  00 00 00 18 18 18 18 18
3CA8  18 18 18 18 18 18 18 18
3CB0  00 00 00 0F 1F 18 18 18
3CB8  18 18 18 1F 1F 18 18 18
3CC0  00 00 00 F8 F8 00 00 00
3CC8  18 18 18 F8 F0 00 00 00
3CD0  00 00 00 FF FF 00 00 00
3CD8  18 18 18 FF FF 00 00 00
3CE0  00 00 00 F0 F8 18 18 18
3CE8  18 18 18 F8 F8 18 18 18
3CF0  00 00 00 FF FF 18 18 18
3CF8  18 18 18 FF FF 18 18 18
3D00  10 38 6C C6 00 00 00 00
3D08  0C 18 30 00 00 00 00 00
3D10  66 66 00 00 00 00 00 00
3D18  3C 66 60 F8 60 66 FE 00
3D20  38 44 BA A2 BA 44 38 00
3D28  7E F4 F4 74 34 34 34 00
3D30  1E 30 38 6C 38 18 F0 00
3D38  18 18 0C 00 00 00 00 00
3D40  40 C0 44 4C 54 1E 04 00
3D48  40 C0 4C 52 44 08 1E 00
3D50  E0 10 62 16 EA 0F 02 00
3D58  00 18 18 7E 18 18 7E 00
3D60  18 18 00 7E 00 18 18 00
3D68  00 00 00 7E 06 06 00 00
3D70  18 00 18 30 66 66 3C 00
3D78  18 00 18 18 18 18 18 00
3D80  00 00 73 DE CC DE 73 00
```

```
3D88  7C C6 C6 FC C6 C6 F8 C0
3D90  00 66 66 3C 66 66 3C 00
3D98  3C 60 60 3C 66 66 3C 00
3DA0  00 00 1E 30 7C 30 1E 00
3DA8  38 6C C6 FE C6 6C 38 00
3DB0  00 C0 60 30 38 6C C6 00
3DB8  00 00 66 66 66 7C 60 60
3DC0  00 00 00 FE 6C 6C 6C 00
3DC8  00 00 00 7E D8 D8 70 00
3DD0  03 06 0C 3C 66 3C 60 C0
3DD8  03 06 0C 66 66 3C 60 C0
3DE0  00 E6 3C 18 38 6C C7 00
3DE8  00 00 66 C3 DB DB 7E 00
3DF0  FE C6 60 30 60 C6 FE 00
3DF8  00 7C C6 C6 C6 6C EE 00
3E00  18 30 60 C0 80 00 00 00
3E08  18 0C 06 03 01 00 00 00
3E10  00 00 00 01 03 06 0C 18
3E18  00 00 00 80 C0 60 30 18
3E20  18 3C 66 C3 81 00 00 00
3E28  18 0C 06 03 03 06 0C 18
3E30  00 00 00 81 C3 66 3C 18
3E38  18 30 60 C0 C0 60 30 18
3E40  18 30 60 C1 83 06 0C 18
3E48  18 0C 06 83 C1 60 30 18
3E50  18 3C 66 C3 C3 66 3C 18
3E58  C3 E7 7E 3C 3C 7E E7 C3
3E60  03 07 0E 1C 38 70 E0 C0
3E68  C0 E0 70 38 1C 0E 07 03
3E70  CC CC 33 33 CC CC 33 33
3E78  AA 55 AA 55 AA 55 AA 55
3E80  FF FF 00 00 00 00 00 00
3E88  03 03 03 03 03 03 03 03
3E90  00 00 00 00 00 00 FF FF
3E98  C0 C0 C0 C0 C0 C0 C0 C0
3EA0  FF FE FC F8 F0 E0 C0 80
3EA8  FF 7F 3F 1F 0F 07 03 01
3EB0  01 03 07 0F 1F 3F 7F FF
3EB8  80 C0 E0 F0 F8 FC FE FF
3EC0  AA 55 AA 55 00 00 00 00
3EC8  0A 05 0A 05 0A 05 0A 05
3ED0  00 00 00 00 AA 55 AA 55
3ED8  A0 50 A0 50 A0 50 A0 50
3EE0  AA 54 A8 50 A0 40 80 00
3EE8  AA 55 2A 15 0A 05 02 01
3EF0  01 02 05 0A 15 2A 55 AA
3EF8  00 80 40 A0 50 A8 54 AA
3F00  7E FF 99 FF BD C3 FF 7E
3F08  7E FF 99 FF C3 BD FF 7E
3F10  38 38 FE FE FE 10 38 00
3F18  10 38 7C FE 7C 38 10 00
3F20  6C FE FE FE 7C 38 10 00
3F28  10 38 7C FE FE 10 38 00
3F30  00 3C 66 C3 C3 66 3C 00
3F38  00 3C 7E FF FF 7E 3C 00
3F40  00 7E 66 66 66 66 7E 00
3F48  00 7E 7E 7E 7E 7E 7E 00
3F50  0F 07 0D 78 CC CC CC 78
```

```
3F58  3C 66 66 66 3C 18 7E 18
3F60  0C 0C 0C 0C 0C 3C 7C 38
3F68  18 1C 1E 1B 18 78 F8 70
3F70  99 5A 24 C3 C3 24 5A 99
3F78  10 38 38 38 38 38 7C D6
3F80  18 3C 7E FF 18 18 18 18
3F88  18 18 18 18 FF 7E 3C 18
3F90  10 30 70 FF FF 70 30 10
3F98  08 0C 0E FF FF 0E 0C 08
3FA0  00 00 18 3C 7E FF FF 00
3FA8  00 00 FF FF 7E 3C 18 00
3FB0  80 E0 F8 FE F8 E0 80 00
3FB8  02 0E 3E FE 3E 0E 02 00
3FC0  38 38 92 7C 10 28 28 28
3FC8  38 38 10 FE 10 28 44 82
3FD0  38 38 12 7C 90 28 24 22
3FD8  38 38 90 7C 12 28 48 88
3FE0  00 3C 18 3C 3C 3C 18 00
3FE8  3C FF FF 18 0C 18 30 18
3FF0  18 3C 7E 18 18 7E 3C 18
3FF8  00 24 66 FF 66 24 00 00
```

6.1.2 Das CPC-464-Basic

Das Basic-ROM des CPC ist nicht so klar unterteilt wie das Betriebssystem. Einheitlich ist hier die Parametrisierung der Routinen, die einen Befehl ausführen. Deshalb wird sie im Listing nicht bei jedem Befehl extra aufgeführt. Einem Befehl wird der Basic-Programmzeiger (Basic-PC), der auf das Zeichen nach dem Befehlstoken zeigt, in HL übergeben. Das Zeichen nach dem Befehlstoken wird im Akku übergeben, und das Zero-Flag ist gesetzt, wenn das Statement direkt nach dem Befehlstoken zu Ende ist. Dies wird von einigen Befehlen zur Fehlererkennung benutzt: Ein RET NZ zu Beginn der Befehlsroutine garantiert, daß der Befehl auch nur dann ausgeführt wird, wenn keine weiteren Zeichen im Statement mehr folgen. Das Carry-Flag ist bei einem Befehls-Ansprung immer gelöscht. Die Interpreterschleife erwartet von der Befehlsroutine in HL den Programmzeiger, der auf das Statementende zeigen muß (":" oder Zeilenende).

Ähnlich wie bei den Befehlen ist auch bei den Basic-Funktionen die Parametrisierung nicht bei jeder Funktion einzeln angegeben. Die Funktionen des Basics sind in drei Gruppen aufgeteilt: Der Gruppe 1 (Tokens $00 bis $1D) wird lediglich das Argument im FAC übergeben. Das Resultat wird ebenfalls wieder im FAC erwartet. Bei Gruppe 2 (Tokens $40 bis $49) wird kein Argument, sondern der Basic-PC in HL übergeben. Zurückgeben müssen die Funktionen der Gruppe 2 das Ergebnis im FAC und den Basic-PC in HL. Für Gruppe 3 (Token $71 bis $7F) gilt die gleiche Parametrisierung wie für Gruppe 2. Der Unterschied zu Gruppe 2 besteht darin, daß vor dem Funktionsaufruf auf eine nach dem Token folgende Klammer auf geprüft und diese überlesen wird.

Obwohl sich für die Operatoren des Basics entsprechende Betrachtungen anstellen ließen, haben wir die Parametrisierung hier jeweils einzeln aufgeführt.

```
**********************************
C000  80                                Kennz. für Vordergrund-ROM
C001  01 00                             Version 1.0
C003  00
C004  4C C0                             Zeiger auf "BASIC"

**********************************    Basic-Kaltstart
                                      IN : DE: LoRAM
                                              HL: HiRAM
C006  31 00 C0    LD    SP,C000        Stackpointer initialisieren
C009  CD CB BC    CALL  BCCB           auf ROM-Erweiterungen prüfen
C00C  CD C4 F4    CALL  F4C4           RAM-Zeiger initialisieren
C00F  DA 00 00    JP    C,0000         kein Platz ? dann Kaltstart
C012  21 00 AC    LD    HL,AC00        Zeiger auf Flag/User-Vektoren
C015  36 00       LD    (HL),00        Flag f. Space-Unterdr. löschen
C017  06 1B       LD    B,1B           Zahl der Bytes
C019  23          INC   HL             Zeiger in User-Vektoren-Tab.
C01A  36 C9       LD    (HL),C9        Opcode für RET in Tabelle
C01C  10 FB       DJNZ  C019           weitere Bytes ?
C01E  21 3F C0    LD    HL,C03F        Zeiger auf " BASIC 1.0"
C021  CD 37 C3    CALL  C337           ausgeben
C024  AF          XOR   A
C025  32 00 AC    LD    (AC00),A       Flag f. Space-Unterdr. löschen
C028  CD CB DD    CALL  DDCB           Direkt-Modus einschalten
C02B  CD 84 CA    CALL  CA84           Fehlernr. und -zeile init.
C02E  CD 97 BD    CALL  BD97           Start-Seed-Wert für RND setzen
C031  CD D3 C0    CALL  C0D3           AUTO abschalten
C034  CD 3E C1    CALL  C13E           NEW-Befehl
C037  11 F0 00    LD    DE,00F0        Nr. des 1. User-Zeichens
C03A  CD 06 F7    CALL  F706           SYMBOL AFTER 240
C03D  18 25       JR    C064           zur Eingabeschleife

**********************************    Meldung des Basics
C03F  20 42 41 53 49 43 20 31             " BASIC 1.0",LF,LF
C047  2E 30 0A 0A 00

**********************************    Name des ROMs
C04C  42 41 53 49 C3 00                   "BASI", "C"+$80

**********************************    Basic-Befehl EDIT
C052  CD E1 CE    CALL  CEE1           Zeilennr. holen
C055  C0          RET   NZ             Statementende ? sonst Fehler
C056  31 00 C0    LD    SP,C000        Stackpointer initialisieren
C059  CD 9A E7    CALL  E79A           Zeile im Programm suchen
C05C  CD 63 E1    CALL  E163           Zeile nach ASCII wandeln
C05F  CD 43 CA    CALL  CA43           und ausgeben, neue Zeile holen
C062  38 54       JR    C,C0B8         kein Abbruch ? dann auswerten

**********************************    Eingabeschleife
C064  CD 01 AC    CALL  AC01           User-Vektor
C067  31 00 C0    LD    SP,C000        Stackpointer initialisieren
C06A  CD 62 C1    CALL  C162           Ausdruckauswertung & I/O init.
C06D  CD D6 DD    CALL  DDD6           Flag für Programm-Modus holen
C070  DC B6 BC    CALL  C,BCB6         Programm ? dann SOUND HOLD
C073  CD 48 BB    CALL  BB48           Break-Taste verriegeln
C076  CD 86 C3    CALL  C386           Bildschirm initialisieren
C079  3A 45 AE    LD    A,(AE45)       Flag für geschütztes Programm
C07C  B7          OR    A                Programm geschützt ?
```

```
C07D  C4 3E C1     CALL   NZ,C13E        dann löschen
C080  3A AA AD     LD     A,(ADAA)      Fehlernr. (ERR)
C083  D6 02        SUB    02
C085  20 09        JR     NZ,C090       nicht Nr. für "Syntax error" ?
C087  32 AA AD     LD     (ADAA),A      sonst Fehlernr. löschen
C08A  CD DF CA     CALL   CADF          Fehlerzeilennr. (ERRL) nach HL
C08D  EB           EX     DE,HL         nach DE
C08E  38 C6        JR     C,C056        Programm-Modus ? dann EDIT
C090  21 CC C0     LD     HL,C0CC       Zeiger auf "Ready"
C093  CD 41 C3     CALL   C341          ausgeben
C096  CD CB DD     CALL   DDCB          Direkt-Modus einschalten

C099  3A 1C AC     LD     A,(AC1C)      Flag für AUTO
C09C  B7           OR     A
C09D  28 11        JR     Z,C0B0        AUTO nicht aktiv ?
C09F  CD 02 C1     CALL   C102          AUTO-Eingabezeile holen
C0A2  30 C0        JR     NC,C064       Abbruch ? dann Eingabeschleife
C0A4  7E           LD     A,(HL)        erstes Zeichen aus Zeile
C0A5  B7           OR     A
C0A6  28 F1        JR     Z,C099        Zeilenende ? dann nächste Zl.
C0A8  CD D2 E6     CALL   E6D2          Zeile im Programm einfügen
C0AB  CD 7A C1     CALL   C17A          Basic-Zeiger initialisieren
C0AE  18 E9        JR     C099          nächste Zeile holen

C0B0  CD 3B CA     CALL   CA3B          Eingabezeile holen
C0B3  30 FB        JR     NC,C0B0       Abbruch ? dann neue Zeile
C0B5  CD 4E C3     CALL   C34E          LF ausgeben
C0B8  CD BC E6     CALL   E6BC          Zeile auswerten, ggf. einfügen
C0BB  30 05        JR     NC,C0C2       Direkteingabe ?
C0BD  C4 7A C1     CALL   NZ,C17A       Leerz. ? sonst Basic-Zg. init.
C0C0  18 D4        JR     C096          nächste Zeile holen

C0C2  CD BB DE     CALL   DEBB          Zeile tokenisieren
C0C5  CD 53 C4     CALL   C453          Break-Taste entriegeln
C0C8  2B           DEC    HL            Zeiger vor Zeile
C0C9  C3 74 DD     JP     DD74          Eingabe ausführen

***********************************
C0CC  52 65 61 64 79 0A 00              Ready..

*********************************** AUTO ausschalten
C0D3  AF           XOR    A               Flag für AUTO ausgeschaltet
C0D4  18 05        JR     C0DB

*********************************** AUTO-Zeilennummer setzen
C0D6  22 1D AC     LD     (AC1D),HL       Zeilennummer
C0D9  3E FF        LD     A,FF            und Flag für AUTO
C0DB  32 1C AC     LD     (AC1C),A        setzen
C0DE  C9           RET

*********************************** Basic-Befehl AUTO
C0DF  11 0A 00     LD     DE,000A         Default-Startzeilennr.
C0E2  28 02        JR     Z,C0E6          Statementende ? dann Default
C0E4  FE 2C        CP     2C              Komma ?
C0E6  C4 E1 CE     CALL   NZ,CEE1         sonst Startzeilennr. holen
C0E9  D5           PUSH   DE              und retten
C0EA  11 0A 00     LD     DE,000A         Default-Schrittweite
C0ED  CD 55 DD     CALL   DD55            folgt Komma ?
```

```
C0F0  DC E1 CE     CALL   C,CEE1        dann Schrittweite holen
C0F3  CD 4A DD     CALL   DD4A          auf Statementende prüfen
C0F6  EB           EX     DE,HL
C0F7  22 1F AC     LD     (AC1F),HL     Schrittweite speichern
C0FA  E1           POP    HL            Startzeilennr.
C0FB  CD D6 C0     CALL   C0D6          als AUTO-Zeilennr. setzen
C0FE  C1           POP    BC            Aufrufadresse vom Stack
C0FF  C3 96 C0     JP     C096          zur Eingabeschleife

*********************************** AUTO-Eingabezeile holen
                                    OUT: CY=0 für Abbruch
                                         HL: Zeiger auf Eingabezeile
C102  2A 1D AC     LD     HL,(AC1D)     aktuelle AUTO-Zeilennr.
C105  E5           PUSH   HL            retten
C106  CD 79 EE     CALL   EE79          und ausgeben
C109  D1           POP    DE            AUTO-Zeilennr.
C10A  CD A3 E7     CALL   E7A3          Zeile im Programm suchen
C10D  3E 2A        LD     A,2A          "*" als Warnung
C10F  38 02        JR     C,C113        Zeile schon vorhanden ?
C111  3E 20        LD     A,20          sonst Space
C113  CD 56 C3     CALL   C356          ausgeben
C116  CD D3 C0     CALL   C0D3          AUTO ausschalten
C119  CD 3B CA     CALL   CA3B          Eingabezeile holen
C11C  D0           RET    NC            Abbruch ? dann zurück
C11D  CD 4E C3     CALL   C34E          LF ausgeben
C120  E5           PUSH   HL            Zeiger auf Zeile
C121  2A 1F AC     LD     HL,(AC1F)     AUTO-Schrittweite
C124  19           ADD    HL,DE         zu Zeilennr. addieren
C125  D4 D6 C0     CALL   NC,C0D6       neu setzen, wenn nicht zu groß
C128  E1           POP    HL
C129  37           SCF                  CY=1 für keinen Abbruch
C12A  C9           RET

*********************************** Basic-Befehl NEW
C12B  C0           RET    NZ            Statementende ? sonst Fehler
C12C  CD 3E C1     CALL   C13E          NEW ausführen
C12F  C3 64 C0     JP     C064          zur Eingabeschleife

*********************************** Basic-Befehl CLEAR
C132  E5           PUSH   HL            Basic-PC retten
C133  CD 8C C1     CALL   C18C          Variablen löschen
C136  CD 5B C1     CALL   C15B          Ausdrucksausw. und I/O init.
C139  CD 7A C1     CALL   C17A          Basic-Zeiger init.
C13C  E1           POP    HL            Basic-PC zurück
C13D  C9           RET

*********************************** NEW Fortsetzung
C13E  2A 7F AE     LD     HL,(AE7F)     Zeiger auf Start d. freien RAM
C141  EB           EX     DE,HL         nach DE
C142  2A 7B AE     LD     HL,(AE7B)     HIMEM-Zeiger
C145  CD DA FF     CALL   FFDA          Differenz (freier Platz) n. BC
C148  62           LD     H,D             Zeiger auf Start des freien
C149  6B           LD     L,E             RAM als Quelladresse nach HL
C14A  13           INC    DE            Zeiger danach als Zieladresse
C14B  AF           XOR    A             Null
C14C  77           LD     (HL),A        an RAM-Start
C14D  ED B0        LDIR                 freies RAM löschen
C14F  32 45 AE     LD     (AE45),A      Flag f. ungeschütztes Programm
```

```
C152  CD 76 E6     CALL   E676        Programm löschen
C155  CD 8C C1     CALL   C18C        Variablen löschen
C158  CD 6B C1     CALL   C16B        Basic initialisieren
C15B  CD AD D2     CALL   D2AD        Kassetten-I/O init.
C15E  AF           XOR    A           Flag für RAD
C15F  CD 73 BD     CALL   BD73        setzen

*********************************  Ausdruckauswertung und I/O init.
C162  CD B3 FB     CALL   FBB3        Stringstackpointer init.
C165  CD FD D9     CALL   D9FD        FN-Listenzeiger löschen
C168  C3 9D C1     JP     C19D        Ein-/Ausgabekanäle init.

*********************************  Basic initialisieren
C16B  CD E6 DD     CALL   DDE6        TROFF-Befehl
C16E  CD D3 C0     CALL   C0D3        AUTO ausschalten
C171  CD F2 F1     CALL   F1F2        ZONE 13
C174  CD 76 E6     CALL   E676        Programm löschen
C177  CD B1 D5     CALL   D5B1        Variablenbereich freigeben

*********************************  Basic-Zeiger initialisieren
C17A  CD D9 CB     CALL   CBD9        ON ERROR ausschalten
C17D  CD AB CB     CALL   CBAB        CONT sperren
C180  CD ED C8     CALL   C8ED        Events initialisieren
C183  CD 8E F5     CALL   F58E        Basic-Stackpointer init.
C186  CD D2 D5     CALL   D5D2        definierte Funktionen löschen
C189  C3 E5 DC     JP     DCE5        RESTORE

*********************************  Variablen löschen
C18C  C5           PUSH   BC
C18D  E5           PUSH   HL
C18E  CD CA F5     CALL   F5CA        Stringbereich freigeben
C191  CD AE D5     CALL   D5AE        Variablenzeiger init.
C194  CD FC D5     CALL   D5FC        DEFREAL A-Z
C197  CD 89 E9     CALL   E989        Variablenoffsets löschen
C19A  E1           POP    HL
C19B  C1           POP    BC
C19C  C9           RET

*********************************  Ein-/Ausgabekanäle initialisieren
C19D  AF           XOR    A           Null
C19E  CD AF C1     CALL   C1AF        als Eingabekanal-Nr. setzen
C1A1  AF           XOR    A           Null

*********************************  neue Streamnummer setzen
                                   IN : A: neue Streamnr.
                                   OUT: A: alte Streamnr.
C1A2  E5           PUSH   HL
C1A3  F5           PUSH   AF          neue Nr. retten
C1A4  FE 08        CP     08          Nr. für Bildschirm ?
C1A6  DC B4 BB     CALL   C,BBB4      dann entspr. Window auswählen
C1A9  F1           POP    AF          neue Nr.
C1AA  21 21 AC     LD     HL,AC21     Zeiger auf aktuelle Streamnr.
C1AD  18 04        JR     C1B3        Streamnr. setzen

*********************************  neue Eingabekanal-Nr. setzen
                                   IN : A: neue Kanalnr.
                                   OUT: A: alte Kanalnr.
C1AF  E5           PUSH   HL
```

```
C1B0  21 22 AC     LD     HL,AC22       Zeiger auf aktuelle Kanalnr.
C1B3  D5           PUSH   DE
C1B4  5F           LD     E,A           neue Nr.
C1B5  7E           LD     A,(HL)        alte Nr. retten
C1B6  73           LD     (HL),E        neue Nr. setzen
C1B7  D1           POP    DE
C1B8  E1           POP    HL
C1B9  C9           RET

**********************************   aktuelle Streamnr. holen
                                     OUT: A: aktuelle Streamnr.
                                          CY=1, wenn Bildschirm
C1BA  3A 21 AC     LD     A,(AC21)      aktuelle Streamnr.
C1BD  FE 08        CP     08            <8 bei Bildschirm
C1BF  C9           RET

**********************************   aktuelle Eingabekanalnr. holen
                                     OUT: A: aktuelle Eingabekanalnr.
                                          CY=1, wenn Bildschirm
C1C0  3A 22 AC     LD     A,(AC22)      aktuelle Eingabekanalnr.
C1C3  FE 09        CP     09            <9 bei Bildschirm/Drucker
C1C5  C9           RET

**********************************   optionale Streamnr. holen/setzen
                                     OUT: A: alte Streamnr.
C1C6  CD E3 C1     CALL   C1E3          optionale Filenr. holen
C1C9  18 D7        JR     C1A2          als aktuelle Streamnr. setzen

**********************************   opt. Eingabekanalnr. holen/setzen
                                     OUT: A: alte Eingabekanalnr.
C1CB  CD E3 C1     CALL   C1E3          optionale Filenr. holen
C1CE  18 DF        JR     C1AF          als Eingabekanalnr. setzen

**********************************   opt. Streamnr. setzen/rücksetzen
C1D0  CD E3 C1     CALL   C1E3          optionale Filenr. holen
C1D3  FE 08        CP     08            >8 ?
C1D5  30 2E        JR     NC,C205       dann "Improper argument"
C1D7  CD A2 C1     CALL   C1A2          Filenr. als Streamnr. setzen
C1DA  C1           POP    BC            Aufrufadresse vom Stack
C1DB  F5           PUSH   AF            alte Streamnr. retten
C1DC  CD F9 FF     CALL   FFF9          aufrufende Routine weiterführ.
C1DF  F1           POP    AF            alte Streamnr.
C1E0  C3 A2 C1     JP     C1A2          wieder als aktuelle Nr. setzen

**********************************   optionale Filenr. holen
                                     OUT: A: Filenr.
C1E3  7E           LD     A,(HL)        Zeichen aus Basic-Text
C1E4  FE 23        CP     23            "#" ?
C1E6  3E 00        LD     A,00          Default-Wert
C1E8  C0           RET    NZ            nicht "#" ? dann Default
C1E9  CD F5 C1     CALL   C1F5          sonst Filenr. holen
C1EC  F5           PUSH   AF            Nr. retten
C1ED  CD 55 DD     CALL   DD55          folgt Komma ?
C1F0  D4 4A DD     CALL   NC,DD4A       sonst Test auf Statementende
C1F3  F1           POP    AF            Streamnr.
C1F4  C9           RET
```

```
*********************************    Filenr. holen
                                     OUT: A: Filenr.
C1F5  CD 37 DD     CALL   DD37          Test auf "#"
C1F8  23                                "#"
C1F9  3E 0A        LD     A,0A          Limit+1 für Filenr.

*********************************    Byte kleiner A holen
                                     IN : A: Limit+1
                                     OUT: A: Byte
C1FB  C5           PUSH   BC
C1FC  D5           PUSH   DE
C1FD  47           LD     B,A           Limit
C1FE  CD 67 CE     CALL   CE67          Byte holen, nach A
C201  B8           CP     B             <B ?
C202  D1           POP    DE
C203  C1           POP    BC
C204  D8           RET    C             dann o.k., zurück
C205  1E 05        LD     E,05          Nr. für "Improper argument"
C207  C3 94 CA     JP     CA94          Fehler ausgeben

*********************************    Basic-Befehl PAPER
C20A  CD D0 C1     CALL   C1D0          opt. Streamnr. setzen/rücks.
C20D  01 96 BB     LD     BC,BB96       Zeiger auf TXT SET PAPER
C210  18 06        JR     C218          Paper-Wert setzen

*********************************    Basic-Befehl PEN
C212  CD D0 C1     CALL   C1D0          opt. Streamnr. setzen/rücks.
C215  01 90 BB     LD     BC,BB90       Zeiger auf TXT SET PEN
C218  CD 4B C2     CALL   C24B          Farbstiftnr. holen
C21B  E5           PUSH   HL
C21C  CD F9 FF     CALL   FFF9          Routine zum Setzen aufrufen
C21F  E1           POP    HL
C220  C9           RET

*********************************    Basic-Befehl BORDER
C221  CD 3C C2     CALL   C23C          2 Farbwerte holen
C224  E5           PUSH   HL
C225  CD 38 BC     CALL   BC38          SCR SET BORDER
C228  E1           POP    HL
C229  C9           RET

*********************************    Basic-Befehl INK
C22A  CD 4B C2     CALL   C24B          Farbstiftnr. holen
C22D  F5           PUSH   AF            und retten
C22E  CD 37 DD     CALL   DD37          Test auf ","
C231  2C                                ","
C232  CD 3C C2     CALL   C23C          2 Farbwerte holen
C235  F1           POP    AF            Farbstiftnr.
C236  E5           PUSH   HL
C237  CD 32 BC     CALL   BC32          SCR SET INK
C23A  E1           POP    HL
C23B  C9           RET

*********************************    2 Farbwerte holen
                                     OUT: B: 1. Farbwert
                                          C: 2. Farbwert
C23C  CD 44 C2     CALL   C244          Farbwert holen
C23F  41           LD     B,C           als 1. Farbwert setzen
```

```
C240  CD 55 DD     CALL   DD55         folgt Komma ?
C243  D0           RET    NC           nein ? dann 2. Wert=1. Wert
C244  3E 20        LD     A,20         Limit+1 für Farbwert
C246  CD FB C1     CALL   C1FB         Byte <32 holen
C249  4F           LD     C,A          als Farbwert nach C
C24A  C9           RET

**********************************   Farbstiftnr. holen
                                     OUT: A: Farbstiftnr.
C24B  3E 10        LD     A,10         Limit+1 für Farbstiftnr.
C24D  18 AC        JR     C1FB         Byte <16 holen

**********************************   Basic-Befehl MODE
C24F  3E 03        LD     A,03         Limit+1 für Mode-Nr.
C251  CD FB C1     CALL   C1FB         Byte <3 als Mode-Nr. holen
C254  E5           PUSH   HL
C255  CD 0E BC     CALL   BC0E         SCR SET MODE
C258  E1           POP    HL
C259  C9           RET

**********************************   Basic-Befehl CLS
C25A  CD D0 C1     CALL   C1D0         opt. Filenr. setzen/rücksetzen
C25D  3E 0C        LD     A,0C         ASCII-Code f. Bildsch. löschen
C25F  C3 6E C3     JP     C36E         Zeichen ausgeben

**********************************   Basic-Funktion VPOS
C262  01 67 C2     LD     BC,C267      Routine für Cursorzeile holen
C265  18 12        JR     C279         Cursorzeile holen, nach FAC

**********************************   Cursorzeile holen
                                     OUT: A: Cursorzeile
C267  3A 21 AC     LD     A,(AC21)     aktuelle Streamnr.
C26A  FE 08        CP     08           Drucker oder Kassette ?
C26C  30 97        JR     NC,C205      dann "Improper argument"
C26E  CD 78 BB     CALL   BB78         Cursorposition holen
C271  CD 87 BB     CALL   BB87         Position ggf. korrigieren
C274  7D           LD     A,L          Cursorzeile
C275  C9           RET

**********************************   Basic-Funktion POS
C276  01 90 C2     LD     BC,C290      Routine für Cursorspalte holen
C279  CD F5 C1     CALL   C1F5         Streamnr. holen
C27C  CD A2 C1     CALL   C1A2         als aktuelle Streamnr. setzen
C27F  F5           PUSH   AF           alte Streamnr. retten
C280  CD 37 DD     CALL   DD37         Test auf ")"
C283  29                               ")"
C284  E5           PUSH   HL           Basic-PC retten
C285  CD F9 FF     CALL   FFF9         Routine aufrufen
C288  CD 0A FF     CALL   FF0A         Position nach FAC
C28B  E1           POP    HL           Basic-PC zurück
C28C  F1           POP    AF           alte Streamnr.
C28D  C3 A2 C1     JP     C1A2         wieder setzen

**********************************   horizontale Position f. I/O holen
                                     OUT: A: Position
C290  3A 21 AC     LD     A,(AC21)     aktuelle Streamnr.
C293  FE 08        CP     08           Drucker ?
C295  CA DF C3     JP     Z,C3DF       dann Druckkopfspalte holen
```

```
C298  3A 25 AC     LD      A,(AC25)       aktuelle Position für Kassette
C29B  D0           RET     NC             Kassette ? dann zurück
C29C  C3 9C C3     JP      C39C           Cursorspalte holen

***********************************     aktuelle Ausgabe-Breite holen
                                        OUT: A: Breite
                                               CY=1, wenn gültig
C29F  3A 21 AC     LD      A,(AC21)       aktuelle Streamnr.
C2A2  FE 08        CP      08             Drucker ?
C2A4  28 0D        JR      Z,C2B3         dann WIDTH holen
C2A6  D0           RET     NC             nicht Bildschirm ? dann zurück
C2A7  D5           PUSH    DE
C2A8  E5           PUSH    HL
C2A9  CD 69 BB     CALL    BB69           Window-Begrenzungen holen
C2AC  7A           LD      A,D            rechte Grenze
C2AD  94           SUB     H              - linke Grenze
C2AE  3C           INC     A              +1 ergibt Breite
C2AF  E1           POP     HL
C2B0  D1           POP     DE
C2B1  37           SCF                    CY=1 für gültig
C2B2  C9           RET
C2B3  3A 24 AC     LD      A,(AC24)       WIDTH laden
C2B6  FE FF        CP      FF             CY=0 wenn $FF (ungültig)
C2B8  C9           RET

***********************************     auf Platz in Zeile prüfen
                                        IN : A: benötigte Zeichenzahl
                                        OUT: CY=0, wenn zu lang für Zeile
C2B9  E5           PUSH    HL
C2BA  CD BF C2     CALL    C2BF           auf Platz in Zeile prüfen
C2BD  E1           POP     HL
C2BE  C9           RET

***********************************     auf Platz in Zeile prüfen
                                        IN : A: benötigte Zeichenzahl
                                        OUT: CY=0, wenn zu lang für Zeile
C2BF  67           LD      H,A            Zahl der Zeichen
C2C0  CD 9F C2     CALL    C29F           Ausgabebreite holen
C2C3  3F           CCF                      keine gültige Breite ?
C2C4  D8           RET     C                dann zurück
C2C5  6F           LD      L,A            Breite
C2C6  CD 90 C2     CALL    C290           akt. Position holen
C2C9  3D           DEC     A              Abstand zu Zeilenanfang
C2CA  37           SCF
C2CB  C8           RET     Z              Zeilenanfang ? dann o.k.
C2CC  84           ADD     H              Zahl der ben. Zeichen addieren
C2CD  3F           CCF                      Übertrag ?
C2CE  D0           RET     NC               dann kein Platz
C2CF  3D           DEC     A
C2D0  BD           CP      L              mit max. Breite vergleichen
C2D1  C9           RET

***********************************     Basic-Befehl LOCATE
C2D2  CD D0 C1     CALL    C1D0           opt. Streamnr. setzen/rücks.
C2D5  CD 27 C3     CALL    C327           Koordinaten holen
C2D8  E5           PUSH    HL             Basic-PC retten
C2D9  EB           EX      DE,HL          Koordinaten nach HL
C2DA  24           INC     H                Korrektur für
```

```
C2DB  2C          INC    L             relative Position
C2DC  CD 75 BB    CALL   BB75         Cursor entsprechend setzen
C2DF  E1          POP    HL           Basic-PC zurück
C2E0  C9          RET

***********************************  Basic-Befehl WINDOW
C2E1  7E          LD     A,(HL)         aktuelles Zeichen
C2E2  FE E7       CP     E7             Token für SWAP ?
C2E4  28 17       JR     Z,C2FD         dann WINDOW SWAP
C2E6  CD D0 C1    CALL   C1D0           opt. Streamnr. setzen/rücks.
C2E9  CD 27 C3    CALL   C327           Spaltengrenzen holen
C2EC  D5          PUSH   DE             und retten
C2ED  CD 37 DD    CALL   DD37           Test auf ","
C2F0  2C                                ","
C2F1  CD 27 C3    CALL   C327           Zeilengrenzen holen
C2F4  E3          EX     (SP),HL        Spaltengrenzen zurück
C2F5  7A          LD     A,D              Spaltengrenzen nach H/D,
C2F6  55          LD     D,L              Zeilengrenzen nach L/E
C2F7  6F          LD     L,A              bringen
C2F8  CD 66 BB    CALL   BB66           als Windowgrenzen setzen
C2FB  E1          POP    HL             Basic-PC zurück
C2FC  C9          RET

***********************************  Basic-Befehl WINDOW SWAP
C2FD  CD 3F DD    CALL   DD3F           SWAP-Token übergehen
C300  CD 12 C3    CALL   C312           1. Window-Nr. holen
C303  48          LD     C,B            nach C
C304  CD 55 DD    CALL   DD55           Test auf Komma
C307  06 00       LD     B,00           Default-Window-Nr.
C309  DC 12 C3    CALL   C,C312         Komma ? dann 2. Window-Nr.
C30C  E5          PUSH   HL             Basic-PC retten
C30D  CD B7 BB    CALL   BBB7           Window-Parameter tauschen
C310  E1          POP    HL             Basic-PC zurück
C311  C9          RET

***********************************  Window-Nr. holen
                                     OUT: B: Window-Nr. (CPC 464)
                                          A: Window-Nr.
                                                 (CPC 664/6128)
C312  3E 08       LD     A,08           Limit+1
C314  CD FB C1    CALL   C1FB           Byte<8 als Window-Nr. holen
C317  47          LD     B,A            nach B
C318  C9          RET

***********************************  Basic-Befehl TAG
C319  CD D0 C1    CALL   C1D0           opt. Streamnr. setzen/rücks.
C31C  3E FF       LD     A,FF           Flag für TAG ON
C31E  18 04       JR     C324           Flag setzen

***********************************  Basic-Befehl TAGOFF
C320  CD D0 C1    CALL   C1D0           opt. Streamnr. setzen/rücks.
C323  AF          XOR    A              Flag für TAGOFF
C324  C3 63 BB    JP     BB63           TAG-Flag setzen

***********************************  Koordinaten holen
                                     OUT: D: 1. Koordinate
                                          E: 2. Koordinate
C327  CD 2F C3    CALL   C32F           1. Koordinate holen
```

```
C32A  53          LD     D,E          nach D
C32B  CD 37 DD    CALL   DD37         Test auf ","
C32E  2C                              ","
C32F  D5          PUSH   DE
C330  CD 6D CE    CALL   CE6D         Byte <>0 holen
C333  D1          POP    DE
C334  5F          LD     E,A          als Koordinate
C335  1D          DEC    E            Korrektur f. absolute Koord.
C336  C9          RET

**********************************  I/O init., String ausgeben
                                    IN : HL: Zeiger auf String
C337  3E 84       LD     A,84         132
C339  32 24 AC    LD     (AC24),A     als WIDTH-Wert setzen
C33C  E5          PUSH   HL
C33D  CD 9D C1    CALL   C19D         Ein-/Ausgabe initialisieren
C340  E1          POP    HL

**********************************  String ausgeben
                                    IN : HL: Zeiger auf String
C341  F5          PUSH   AF
C342  E5          PUSH   HL
C343  7E          LD     A,(HL)       Zeichen aus String
C344  23          INC    HL           Zeiger auf nächstes Zeichen
C345  B7          OR     A
C346  C4 56 C3    CALL   NZ,C356      kein Ende ? dann ausgeben
C349  20 F8       JR     NZ,C343      kein Ende ? dann weiter
C34B  E1          POP    HL
C34C  F1          POP    AF
C34D  C9          RET

**********************************  Linefeed ausgeben
C34E  F5          PUSH   AF
C34F  3E 0A       LD     A,0A         ASCII-Code für Linefeed
C351  CD 56 C3    CALL   C356         Zeichen ausgeben
C354  F1          POP    AF
C355  C9          RET

**********************************  Zeichen ausgeben
                                    IN : A: Zeichen
C356  F5          PUSH   AF
C357  CD 5C C3    CALL   C35C         Zeichen ausgeben
C35A  F1          POP    AF
C35B  C9          RET

**********************************  Zeichen ausgeben
                                    IN : A: Zeichen
C35C  FE 0A       CP     0A           Linefeed ?
C35E  20 0E       JR     NZ,C36E      sonst ausgeben
C360  3A 21 AC    LD     A,(AC21)     aktuelle Streamnr.
C363  FE 08       CP     08           Drucker ?
C365  CA A8 C3    JP     Z,C3A8       dann LF an Drucker
C368  D2 EA C3    JP     NC,C3EA      Kassette ?
C36B  C3 92 C3    JP     C392         sonst LF an Bildschirm
C36E  F5          PUSH   AF
C36F  C5          PUSH   BC
C370  4F          LD     C,A          Zeichen
C371  CD 77 C3    CALL   C377         ausgeben
```

```
C374  C1          POP    BC
C375  F1          POP    AF
C376  C9          RET

***********************************  Zeichen ausgeben (ohne LF-Beh.)
                                     IN : C: Zeichen
C377  3A 21 AC    LD     A,(AC21)       aktuelle Streamnr.
C37A  FE 08       CP     08             Drucker ?
C37C  CA B5 C3    JP     Z,C3B5         dann Zeichen an Drucker
C37F  D2 F8 C3    JP     NC,C3F8        Kassette ?
C382  79          LD     A,C            sonst Zeichen
C383  C3 99 C3    JP     C399           auf Bildschirm ausgeben

***********************************  Bildschirm initialisieren
C386  AF          XOR    A              Null
C387  CD 63 BB    CALL   BB63           TAGOFF
C38A  CD 54 BB    CALL   BB54           TXT VDU ENABLE
C38D  CD 9C C3    CALL   C39C           Cursorspalte holen
C390  3D          DEC    A              am Zeilenanfang ?
C391  C8          RET    Z              dann zurück

***********************************  Linefeed auf Bildschirm ausgeben
C392  3E 0D       LD     A,0D           CR
C394  CD 99 C3    CALL   C399           ausgeben
C397  3E 0A       LD     A,0A           LF
C399  C3 5A BB    JP     BB5A           ausgeben

***********************************  Cursorspalte holen
C39C  C5          PUSH   BC
C39D  E5          PUSH   HL
C39E  CD 78 BB    CALL   BB78           Cursorposition holen
C3A1  CD 87 BB    CALL   BB87           ggf. korrigieren
C3A4  7C          LD     A,H            Cursorspalte
C3A5  E1          POP    HL
C3A6  C1          POP    BC
C3A7  C9          RET

***********************************  Linefeed an Drucker ausgeben
C3A8  C5          PUSH   BC
C3A9  0E 0D       LD     C,0D           CR
C3AB  CD B5 C3    CALL   C3B5           an Drucker ausgeben
C3AE  0E 0A       LD     C,0A           LF
C3B0  CD B5 C3    CALL   C3B5           an Drucker ausgeben
C3B3  C1          POP    BC
C3B4  C9          RET

***********************************  Zeichen an Drucker ausgeben
                                     IN : C: Zeichen
C3B5  E5          PUSH   HL
C3B6  79          LD     A,C            Zeichen
C3B7  EE 0D       XOR    0D             CR ?
C3B9  28 13       JR     Z,C3CE         dann Druckkopfposition =1
C3BB  79          LD     A,C            Zeichen
C3BC  FE 20       CP     20             Steuerzeichen ?
C3BE  38 14       JR     C,C3D4         dann Position nicht erhöhen
C3C0  2A 23 AC    LD     HL,(AC23)      Position/WIDTH-Wert
C3C3  24          INC    H              Druckkopfposition erhöhen
C3C4  7D          LD     A,L            Breite
```

```
C3C5  28 07       JR     Z,C3CE       Übertrag bei Position ?
C3C7  BC          CP     H            Position m. Breite vergleichen
C3C8  CC A8 C3    CALL   Z,C3A8       gleich ? dann LF an Drucker
C3CB  3A 23 AC    LD     A,(AC23)     Druckkopfposition
C3CE  3C          INC    A            erhöhen
C3CF  28 03       JR     Z,C3D4       ungültig ?
C3D1  32 23 AC    LD     (AC23),A     sonst neue Position speichern
C3D4  E1          POP    HL
C3D5  79          LD     A,C          Zeichen
C3D6  CD 2B BD    CALL   BD2B         an Drucker ausgeben
C3D9  D8          RET    C            Zeichen ausgegeben ?
C3DA  CD 3C C4    CALL   C43C         sonst auf ESC-Taste prüfen
C3DD  18 F6       JR     C3D5         und warten

**********************************  Druckkopfposition holen, nach A
C3DF  3A 23 AC    LD     A,(AC23)
C3E2  C9          RET

**********************************  Basic-Befehl WIDTH
C3E3  CD 6D CE    CALL   CE6D         Byte <>0 holen
C3E6  32 24 AC    LD     (AC24),A     als WIDTH-Wert setzen
C3E9  C9          RET

**********************************  Linefeed an Kassette ausgeben
C3EA  3E 01       LD     A,01         Eins für Zeilenanfang
C3EC  32 25 AC    LD     (AC25),A     als Kassetten-Position setzen
C3EF  3E 0D       LD     A,0D         CR
C3F1  CD 0D C4    CALL   C40D         an Kassette ausgeben
C3F4  3E 0A       LD     A,0A         LF
C3F6  18 15       JR     C40D         an Kassette ausgeben

**********************************  Zeichen an Kassette ausgeben
                                    IN : C: Zeichen
C3F8  E5          PUSH   HL
C3F9  21 25 AC    LD     HL,AC25      Zeiger auf Kassettenposition
C3FC  79          LD     A,C          Zeichen
C3FD  06 01       LD     B,01         Position für Zeilenanfang
C3FF  FE 0D       CP     0D           CR ?
C401  28 08       JR     Z,C40B       dann Position setzen
C403  FE 20       CP     20           Steuerzeichen ?
C405  38 05       JR     C,C40C       dann Position nicht erhöhen
C407  46          LD     B,(HL)       Position
C408  04          INC    B            erhöhen
C409  28 01       JR     Z,C40C       ungültig ?
C40B  70          LD     (HL),B       sonst wieder speichern
C40C  E1          POP    HL
C40D  CD 95 BC    CALL   BC95         Zeichen an Kassette ausgeben
C410  D8          RET    C            kein Abbruch ? dann zurück
C411  C3 6B CB    JP     CB6B         Break ausgeben, Abbruch

**********************************  Zeichen zurück in Kassettenbuffer
                                    IN : A: Zeichen
C414  C3 86 BC    JP     BC86         CAS RETURN

**********************************  Basic-Funktion EOF
C417  E5          PUSH   HL           Basic-PC retten
C418  CD 89 BC    CALL   BC89         CAS TEST EOF
C41B  28 F4       JR     Z,C411       Abbruch ? dann behandeln
```

```
C41D  3F          CCF
C41E  9F          SBC   A          A=$FF bei EOF, sonst A=0
C41F  CD 05 FF    CALL  FF05       Byte nach FAC
C422  E1          POP   HL         Basic-PC zurück
C423  C9          RET

*********************************** Zeichen einlesen
                                  OUT: A: Zeichen
C424  3A 22 AC    LD    A,(AC22)     aktueller Eingabekanal
C427  FE 09       CP    09           Kassette ?
C429  CA 80 BC    JP    Z,BC80       dann CAS IN CHAR
C42C  CD 09 BB    CALL  BB09         Zeichen von Tastatur holen
C42F  D8          RET   C            Taste gedrückt ? dann fertig
C430  CD 81 BB    CALL  BB81         Cursor einschalten
C433  CD 06 BB    CALL  BB06         auf Taste warten
C436  C3 84 BB    JP    BB84         Cursor wieder ausschalten

*********************************** Zeichen von Tastatur holen
C439  C3 09 BB    JP    BB09         KM READ CHAR

*********************************** auf ESC-Taste prüfen
C43C  CD 09 BB    CALL  BB09         Zeichen von Tastatur holen
C43F  D0          RET   NC           keine Taste gedrückt ?
C440  FE FC       CP    FC           ESC-Taste ?
C442  C0          RET   NZ           nein ? dann zurück
C443  C5          PUSH  BC
C444  D5          PUSH  DE
C445  E5          PUSH  HL
C446  CD 6F C4    CALL  C46F         auf weitere Taste warten
C449  DA 6B CB    JP    C,CB6B       Abbruch ? dann Break ausgeben
C44C  CD 53 C4    CALL  C453         Abbruch durch Break ermögl.
C44F  E1          POP   HL
C450  D1          POP   DE
C451  C1          POP   BC
C452  C9          RET

*********************************** ESC-Abbruch einmal ermöglichen
C453  E5          PUSH  HL
C454  11 5E C4    LD    DE,C45E      Routinenadresse
C457  0E FD       LD    C,FD         ROM-Konfig., Basic-ROM ein
C459  CD 45 BB    CALL  BB45         KM ARM BREAK
C45C  E1          POP   HL
C45D  C9          RET

*********************************** Break-Event-Routine
C45E  E5          PUSH  HL           Zeiger auf Routinenadresse
C45F  CD 09 BB    CALL  BB09         Zeichen von Tastatur einlesen
C462  30 04       JR    NC,C468      keine Taste gedrückt ?
C464  FE EF       CP    EF           Break-Event durch ESC ?
C466  20 F7       JR    NZ,C45F      nein ? d. weitere Tasten lesen
C468  CD 6F C4    CALL  C46F         auf weitere Taste warten
C46B  E1          POP   HL           Zeiger auf Routinenadresse
C46C  C3 47 C8    JP    C847         Flag f. Abbruch o. ON BREAK s.

*********************************** nach ESC/Break auf Taste warten
                                  OUT: CY=1 bei Abbruch
C46F  CD B6 BC    CALL  BCB6         SOUND HOLD
C472  F5          PUSH  AF           Flag für aktiv retten
```

```
C473  CD 30 C4    CALL   C430        Taste einlesen
C476  FE EF       CP     EF          Code f. Break-Ev. durch ESC ?
C478  28 F9       JR     Z,C473      dann neue Taste holen
C47A  FE FC       CP     FC          ESC-Taste ?
C47C  28 0B       JR     Z,C489      dann Flag für Abbruch setzen
C47E  FE 20       CP     20          Space ?
C480  C4 0C BB    CALL   NZ,BB0C     nein ? dann zurück in Buffer
C483  F1          POP    AF          Sound aktiv ?
C484  DC B9 BC    CALL   C,BCB9      dann SOUND CONTINUE
C487  B7          OR     A           CY=0 für keinen Abbruch
C488  C9          RET
C489  F1          POP    AF
C48A  37          SCF                CY=1 für Abbruch
C48B  C9          RET

*********************************** Basic-Befehl ORIGIN
C48C  CD 1A C5    CALL   C51A        Koordinaten holen
C48F  C5          PUSH   BC            als Graphikcursorposition
C490  D5          PUSH   DE            retten
C491  CD 55 DD    CALL   DD55        folgt Komma ?
C494  30 18       JR     NC,C4AE     nein ?
C496  CD 1A C5    CALL   C51A        sonst Koordinaten holen
C499  C5          PUSH   BC            als Grenzen (links,rechts)
C49A  D5          PUSH   DE            retten
C49B  CD 37 DD    CALL   DD37        Test auf ","
C49E  2C                             ","
C49F  CD 1A C5    CALL   C51A        Koordinaten holen
C4A2  C5          PUSH   BC          untere Grenze retten,
C4A3  E3          EX     (SP),HL     nach HL, Basic-PC retten
C4A4  CD D2 BB    CALL   BBD2        Grenzen oben/unten setzen
C4A7  E1          POP    HL          Basic-PC vom Stack
C4A8  D1          POP    DE          linke Grenze vom Stack,
C4A9  E3          EX     (SP),HL     rechte nach HL, PC retten
C4AA  CD CF BB    CALL   BBCF        Grenzen links/rechts setzen
C4AD  E1          POP    HL          Basic-PC vom Stack
C4AE  D1          POP    DE          X-Cursorposition vom Stack
C4AF  E3          EX     (SP),HL     Y-Pos. nach HL, PC retten
C4B0  CD C9 BB    CALL   BBC9        Koordinaten-Ursprung setzen
C4B3  E1          POP    HL          Basic-PC vom Stack
C4B4  C9          RET

*********************************** Basic-Befehl CLG
C4B5  CD 51 DD    CALL   DD51        Statementende ?
C4B8  38 06       JR     C,C4C0      dann keinen Farbstift holen
C4BA  CD 4B C2    CALL   C24B        Farbstiftnr. holen
C4BD  CD E4 BB    CALL   BBE4        GRA SET PAPER
C4C0  E5          PUSH   HL
C4C1  CD DB BB    CALL   BBDB        Graphik-Window löschen
C4C4  E1          POP    HL
C4C5  C9          RET

*********************************** Basic-Befehl DRAW
C4C6  01 F6 BB    LD     BC,BBF6     GRA LINE ABSOLUTE
C4C9  18 0D       JR     C4D8

*********************************** Basic-Befehl DRAWR
C4CB  01 F9 BB    LD     BC,BBF9     GRA LINE RELATIVE
C4CE  18 08       JR     C4D8
```

```
*********************************** Basic-Befehl PLOT
C4D0  01 EA BB     LD    BC,BBEA     GRA PLOT ABSOLUTE
C4D3  18 03        JR    C4D8

*********************************** Basic-Befehl PLOTR
C4D5  01 ED BB     LD    BC,BBED     GRA PLOT RELATIVE
C4D8  C5           PUSH  BC          Routinenadresse retten
C4D9  CD 1A C5     CALL  C51A        Koordinaten holen
C4DC  CD 55 DD     CALL  DD55        folgt Komma ?
C4DF  30 06        JR    NC,C4E7     nein ?
C4E1  CD 4B C2     CALL  C24B        sonst Farbstift-Nr. holen
C4E4  CD DE BB     CALL  BBDE        GRA SET PEN
C4E7  18 28        JR    C511        Routine ausführen

*********************************** Basic-Befehl TEST
C4E9  01 F0 BB     LD    BC,BBF0     GRA TEST ABSOLUTE
C4EC  18 03        JR    C4F1

*********************************** Basic-Funktion TESTR
C4EE  01 F3 BB     LD    BC,BBF3     GRA TEST RELATIVE
C4F1  C5           PUSH  BC          Routinenadresse retten
C4F2  CD 1A C5     CALL  C51A        Koordinaten holen
C4F5  CD 37 DD     CALL  DD37        Test auf ")"
C4F8  29                             ")"
C4F9  E3           EX    (SP),HL     Routinenadr. v. St., PC retten
C4FA  C5           PUSH  BC          Y-Koordinate auf Stack
C4FB  E3           EX    (SP),HL     nach HL, Routinenadr. retten
C4FC  C1           POP   BC          und nach BC
C4FD  CD F9 FF     CALL  FFF9        Routine anspringen
C500  CD 0A FF     CALL  FF0A        Funktionsergebnis in FAC
C503  E1           POP   HL          Basic-PC zurück
C504  C9           RET

*********************************** Basic-Befehl MOVE
C505  01 C0 BB     LD    BC,BBC0     GRA MOVE ABSOLUTE
C508  18 03        JR    C50D

*********************************** Basic-Befehl MOVER
C50A  01 C3 BB     LD    BC,BBC3     GRA MOVE RELATIVE
C50D  C5           PUSH  BC          Routinenadresse retten
C50E  CD 1A C5     CALL  C51A        Koordinaten holen
C511  E3           EX    (SP),HL     Routinenadr. v. St., PC retten
C512  C5           PUSH  BC          Y-Koordinate auf Stack
C513  E3           EX    (SP),HL     nach HL, Routinenadr. retten
C514  C1           POP   BC          und nach BC
C515  CD F9 FF     CALL  FFF9        Routine anspringen
C518  E1           POP   HL          Basic-PC zurück
C519  C9           RET

*********************************** Graphik-Koordinaten holen
                                     OUT: DE: 1. Koordinate
                                          BC: 2. Koordinate
C51A  CD 86 CE     CALL  CE86        Integerwert holen
C51D  D5           PUSH  DE          und retten
C51E  CD 37 DD     CALL  DD37        Test auf ","
C521  2C                             ","
C522  CD 86 CE     CALL  CE86        Integerwert holen
C525  42           LD    B,D           nach
```

```
C526  4B          LD     C,E             BC
C527  D1          POP    DE            1. Koordinate vom Stack
C528  C9          RET

***********************************  Basic-Befehl FOR
C529  CD B3 D6    CALL   D6B3          einfache Variable holen
C52C  E5          PUSH   HL            Basic-PC retten
C52D  C5          PUSH   BC            Typflag der Variablen
C52E  D5          PUSH   DE            und Variablenadresse retten
C52F  CD C5 C9    CALL   C9C5          zugehöriges NEXT suchen
C532  22 2C AC    LD     (AC2C),HL     Zeiger nach NEXT-Token
C535  D5          PUSH   DE            Adresse der FOR-Zeile und
C536  E5          PUSH   HL            Zeiger nach NEXT-Token retten
C537  EB          EX     DE,HL         und nach DE
C538  CD 32 C6    CALL   C632          offene Schleife m. glei. NEXT
C53B  CC AC F5    CALL   Z,F5AC        gefunden ? dann vom Stack
C53E  E1          POP    HL            Zeiger nach NEXT-Token
C53F  CD 51 DD    CALL   DD51          Statementende ?
C542  11 00 00    LD     DE,0000       Kennz. f. keine NEXT-Variable
C545  D4 86 D6    CALL   NC,D686       nein ? dann NEXT-Var. holen
C548  44          LD     B,H             Zeiger nach NEXT-Variable
C549  4D          LD     C,L             bzw. NEXT-Token nach HL
C54A  E1          POP    HL              FOR-Variablen-Adresse
C54B  E3          EX     (SP),HL         vom Stack
C54C  7A          LD     A,D             NEXT-Variable
C54D  B3          OR     E               vorhanden ?
C54E  C4 B8 FF    CALL   NZ,FFB8       Variablenadr. ggf. gleich ?
C551  C2 F6 C5    JP     NZ,C5F6       nein ? dann "Unexpected NEXT"
C554  EB          EX     DE,HL         FOR-Variablenadresse nach DE
C555  CD D2 DD    CALL   DDD2          akt. (NEXT-)Zeilenadr. holen
C558  E3          EX     (SP),HL       retten, FOR-Zeilenadresse
C559  CD CE DD    CALL   DDCE          wieder als akt. Zeilenadr.
C55C  E1          POP    HL            NEXT-Zeilenadresse
C55D  F1          POP    AF            Typflag der FOR-Variablen
C55E  E3          EX     (SP),HL       FOR-Zeilenadr. retten, PC zur.
C55F  D5          PUSH   DE            FOR-Variablenadresse,
C560  C5          PUSH   BC            Zeiger nach NEXT-Statement
C561  E5          PUSH   HL            und PC nach FOR-Var. retten
C562  01 05 16    LD     BC,1605       Stack-Kennz./Typ für REAL
C565  B9          CP     C             mit Variablentyp vergleichen
C566  28 0B       JR     Z,C573        REAL ?
C568  01 02 10    LD     BC,1002       Stack-Kennz./Typ für INTEGER
C56B  B9          CP     C             mit Variablentyp vergleichen
C56C  28 05       JR     Z,C573        INTEGER ?
C56E  1E 0D       LD     E,0D          sonst Nr. f. "Type mismatch"
C570  C3 94 CA    JP     CA94          Fehler ausgeben
C573  78          LD     A,B           Größe des Stackeintrags
C574  CD B0 F5    CALL   F5B0          Platz auf Basic-Stack reserv.
C577  73          LD     (HL),E
C578  23          INC    HL              FOR-Variablenadresse
C579  72          LD     (HL),D          auf Basic-Stack eintragen
C57A  23          INC    HL
C57B  E3          EX     (SP),HL       Eintragszeiger retten, PC zur.
C57C  CD 37 DD    CALL   DD37          Test auf "="
C57F  EF                               "="
C580  CD FB CE    CALL   CEFB          Startwert holen
C583  79          LD     A,C           FOR-Variablentyp
C584  CD D7 FE    CALL   FED7          Startwert-Typ angleichen
```

```
C587  E5          PUSH   HL          Basic-PC retten
C588  21 27 AC    LD     HL,AC27     Zeiger auf Zwischenspeicher
C58B  CD 62 FF    CALL   FF62        Startwert zwischenspeichern
C58E  E1          POP    HL          Basic-PC
C58F  CD 37 DD    CALL   DD37        Test auf TO
C592  EC                             Token für TO
C593  CD FB CE    CALL   CEFB        Endwert holen
C596  E3          EX     (SP),HL     PC retten, Eintragszeiger zur.
C597  79          LD     A,C         Typ der FOR-Variablen
C598  CD D7 FE    CALL   FED7        Endwert-Typ angleichen
C59B  CD 62 FF    CALL   FF62        Endwert auf Basic-Stack
C59E  EB          EX     DE,HL         Eintragszeiger
C59F  E3          EX     (SP),HL       retten, Basic-PC
C5A0  EB          EX     DE,HL         zurück
C5A1  21 01 00    LD     HL,0001     Default-Stepwert
C5A4  CD 0D FF    CALL   FF0D        in FAC eintragen
C5A7  EB          EX     DE,HL       Basic-PC nach HL
C5A8  7E          LD     A,(HL)      Zeichen nach Endwert
C5A9  FE E6       CP     E6          Token für STEP ?
C5AB  20 06       JR     NZ,C5B3     nein ? dann Default
C5AD  CD 3F DD    CALL   DD3F        sonst STEP-Token übergehen
C5B0  CD FB CE    CALL   CEFB        Stepwert holen
C5B3  79          LD     A,C         Typflag der FOR-Variablen
C5B4  CD D7 FE    CALL   FED7        Stepwert-Typ angleichen
C5B7  E3          EX     (SP),HL     PC retten, Eintragszeiger zur.
C5B8  CD 62 FF    CALL   FF62        Stepwert auf Basic-Stack
C5BB  CD A3 FD    CALL   FDA3        Vorzeichen des Stepwerts holen
C5BE  EB          EX     DE,HL       Eintragszeiger nach HL
C5BF  77          LD     (HL),A      Vorzeichen auf Basic-Stack
C5C0  23          INC    HL          Eintragszeiger
C5C1  EB          EX     DE,HL       nach DE
C5C2  E1          POP    HL          Basic-PC zurück
C5C3  CD 4A DD    CALL   DD4A        Test auf Statementende
C5C6  EB          EX     DE,HL       PC nach DE, Eintragsz. nach HL
C5C7  73          LD     (HL),E        Basic-PC
C5C8  23          INC    HL            (nach FOR-Statement)
C5C9  72          LD     (HL),D        auf Basic-Stack eintragen
C5CA  23          INC    HL
C5CB  EB          EX     DE,HL         Adresse des
C5CC  CD D2 DD    CALL   DDD2          FOR-Zeilenanfangs
C5CF  EB          EX     DE,HL         nach DE
C5D0  73          LD     (HL),E
C5D1  23          INC    HL            FOR-Zeilenadresse
C5D2  72          LD     (HL),D        auf Basic-Stack eintragen
C5D3  23          INC    HL
C5D4  D1          POP    DE          Zeiger nach NEXT-Statement
C5D5  73          LD     (HL),E
C5D6  23          INC    HL            auf Basic-Stack
C5D7  72          LD     (HL),D        eintragen
C5D8  23          INC    HL
C5D9  ED 5B 2C AC LD     DE,(AC2C)   Zeiger nach NEXT-Token
C5DD  73          LD     (HL),E
C5DE  23          INC    HL            auf Basic-Stack
C5DF  72          LD     (HL),D        eintragen
C5E0  23          INC    HL
C5E1  70          LD     (HL),B      Länge des Eintrags eintragen
C5E2  D1          POP    DE          Adresse der FOR-Variablen
C5E3  21 27 AC    LD     HL,AC27     Adresse des Startwertes
```

```
C5E6  CD 66 FF     CALL   FF66         Startwert an Variable zuweisen
C5E9  AF           XOR    A              Flag für Test auf
C5EA  32 26 AC     LD     (AC26),A       Schleifenende setzen
C5ED  E1           POP    HL           FOR-Zeilenadresse
C5EE  CD CE DD     CALL   DDCE         als aktuelle Zeilenadr. setzen
C5F1  2A 2C AC     LD     HL,(AC2C)    Zeiger nach NEXT-Token als PC
C5F4  18 0A        JR     C600         auf Schleifenende testen

C5F6  1E 01        LD     E,01         Nr. für "Unexpected NEXT"
C5F8  C3 94 CA     JP     CA94         Fehler ausgeben

**********************************  Basic-Befehl NEXT
C5FB  3E FF        LD     A,FF           Flag für Schleifendurchlauf
C5FD  32 26 AC     LD     (AC26),A     setzen
C600  EB           EX     DE,HL        Zeiger nach NEXT-Token nach DE
C601  CD 32 C6     CALL   C632         zugehörige FOR-Schleife suchen
C604  20 F0        JR     NZ,C5F6      gef. ? sonst "Unexpected NEXT"
C606  EB           EX     DE,HL        Zeiger oberh. d. Eint. nach HL
C607  CD AC F5     CALL   F5AC         darüberliegende Schlf. löschen
C60A  EB           EX     DE,HL        Zeiger oberh. d. nächst. Eint.
C60B  E5           PUSH   HL           = Zeiger auf diesen Eintrag
C60C  CD 61 C6     CALL   C661         Stepw. ggf. addieren, Ende pr.
C60F  28 0F        JR     Z,C620       Schleifenende ?
C611  F1           POP    AF           Zeiger auf Stackeintr. löschen
C612  23           INC    HL           Zeiger nach Stepwert-Vorzeich.
C613  5E           LD     E,(HL)
C614  23           INC    HL             Zeiger nach FOR-Statement
C615  56           LD     D,(HL)         nach DE
C616  23           INC    HL
C617  7E           LD     A,(HL)
C618  23           INC    HL             FOR-Zeilenadresse
C619  66           LD     H,(HL)         nach HL
C61A  6F           LD     L,A
C61B  CD CE DD     CALL   DDCE         als aktuelle Zeilenadr. setzen
C61E  EB           EX     DE,HL        Zg. nach FOR-Statement als PC
C61F  C9           RET
C620  01 05 00     LD     BC,0005      Offset zu NEXT-Zeiger
C623  09           ADD    HL,BC        zu Zeiger auf Vorzeichen add.
C624  5E           LD     E,(HL)         Zeiger nach NEXT-
C625  23           INC    HL             Variable bzw.
C626  56           LD     D,(HL)         NEXT-Token
C627  E1           POP    HL           Zeiger auf diesen Eintrag
C628  CD AC F5     CALL   F5AC         Schleife v. Basic-Stack lösch.
C62B  EB           EX     DE,HL        Zg. nach NEXT-Statem. als PC
C62C  CD 55 DD     CALL   DD55         folgt Komma ?
C62F  38 CF        JR     C,C600       dann nächste NEXT-Variable
C631  C9           RET

**********************************  offene FOR-Schleife suchen
                                    IN : DE: Zeiger nach zugeh. NEXT
                                    OUT: DE: Zeiger für Eintrag
                                             HL: Zeiger f. nächst. Eintr.
                                             A,B: Länge des Eintrags
                                             Z=1, wenn gefunden
C632  2A 8B B0     LD     HL,(B08B)    Basic-Stackpointer
C635  E5           PUSH   HL           retten
C636  2B           DEC    HL           Zeiger auf obersten Eintrag
C637  46           LD     B,(HL)       Länge des obersten Eintrags
```

```
C638  23         INC   HL          alten Zeiger wieder zurück
C639  7D         LD    A,L
C63A  90         SUB   B             Länge subtrahieren,
C63B  6F         LD    L,A           gibt Zeiger für
C63C  9F         SBC   A             nächsten Eintrag
C63D  84         ADD   H
C63E  67         LD    H,A
C63F  E3         EX    (SP),HL     retten, alten Zeiger zurück
C640  78         LD    A,B         Länge
C641  FE 07      CP    07          WHILE-Schleife ?
C643  28 19      JR    Z,C65E      dann weiter suchen
C645  FE 10      CP    10          Integer-FOR-Schleife ?
C647  28 04      JR    Z,C64D      dann auswerten
C649  FE 16      CP    16          REAL-FOR-Schleife ?
C64B  20 0D      JR    NZ,C65A     nein ? dann nichts gefunden
C64D  E5         PUSH  HL          Zeiger für diesen Stack-Eintr.
C64E  2B         DEC   HL            Zeiger auf zugehörigen
C64F  2B         DEC   HL            NEXT-Zeiger
C650  7E         LD    A,(HL)
C651  2B         DEC   HL            Zeiger nach zugehörigem
C652  6E         LD    L,(HL)        NEXT-Token laden
C653  67         LD    H,A
C654  CD B8 FF   CALL  FFB8        mit Zg. nach ges. NEXT vergl.
C657  E1         POP   HL          Zeiger für Stackeintrag
C658  20 04      JR    NZ,C65E     ungleich ? dann weiter suchen
C65A  EB         EX    DE,HL       Stack-Suchzeiger nach DE
C65B  E1         POP   HL          Zeiger für nächsten Eintrag
C65C  78         LD    A,B         Länge des Eintrags
C65D  C9         RET
C65E  E1         POP   HL          Zeiger für nächsten Eintrag
C65F  18 D4      JR    C635        weiter suchen

*********************************  Stepw. ggf. addieren, Ende prüfen
                                   IN : HL: Zeiger auf Stackeintrag
                                         A: Eintragsgröße
                                   OUT: A=0, Z=1 bei Schleifenende
                                        HL: Zeiger auf Stepw.-Vorz.

C661  5E         LD    E,(HL)
C662  23         INC   HL          FOR-Variablenadresse
C663  56         LD    D,(HL)      aus Eintrag
C664  23         INC   HL
C665  FE 10      CP    10          Eintragsgröße für Integer ?
C667  28 2D      JR    Z,C696      dann Integer-Schleife
C669  E5         PUSH  HL          Eintragszeiger retten
C66A  01 05 00   LD    BC,0005     Eintragsgröße
C66D  79         LD    A,C         Typflag für REAL
C66E  EB         EX    DE,HL       FOR-Variablenadresse nach HL
C66F  CD 4B FF   CALL  FF4B        FOR-Variable in FAC holen
C672  E1         POP   HL          Eintragszeiger
C673  3A 26 AC   LD    A,(AC26)    Flag für Test/Durchlauf
C676  B7         OR    A
C677  28 10      JR    Z,C689      nur Test auf Ende ?
C679  E5         PUSH  HL          sonst Eintragszeiger retten
C67A  09         ADD   HL,BC       Endw.-Gr. add., gibt Stepw.-Z.
C67B  CD CC FC   CALL  FCCC        Stepwert zu FAC addieren
C67E  E1         POP   HL          Zeiger auf Endwert
C67F  E5         PUSH  HL
C680  2B         DEC   HL
```

```
C681  56          LD    D,(HL)      FOR-Variablenadresse
C682  2B          DEC   HL          laden
C683  5E          LD    E,(HL)
C684  EB          EX    DE,HL      nach HL
C685  CD 62 FF    CALL  FF62       FAC wieder nach FOR-Variable
C688  E1          POP   HL         Zeiger auf Endwert
C689  E5          PUSH  HL
C68A  0E 05       LD    C,05       Typflag für REAL
C68C  CD 09 FD    CALL  FD09       FAC mit Endwert vergleichen
C68F  E1          POP   HL         Zeiger auf Endwert
C690  01 0A 00    LD    BC,000A    Step- und Endwert übergehen
C693  09          ADD   HL,BC      gibt Zeiger auf Stepwert-Vorz.
C694  96          SUB   (HL)       vom Vergleichsergebnis abz.
C695  C9          RET

C696  E5          PUSH  HL         Endwertzeiger retten
C697  EB          EX    DE,HL      FOR-Variablenadresse nach HL
C698  5E          LD    E,(HL)
C699  23          INC   HL           Variablenwert nach DE
C69A  56          LD    D,(HL)
C69B  3A 26 AC    LD    A,(AC26)   Flag für Test/Durchlauf
C69E  B7          OR    A
C69F  28 16       JR    Z,C6B7     nur Test auf Ende ?
C6A1  E3          EX    (SP),HL    FOR-Variablenadresse retten
C6A2  E5          PUSH  HL         Eintragszeiger wieder retten
C6A3  23          INC   HL
C6A4  23          INC   HL         Zeiger auf Stepwert
C6A5  7E          LD    A,(HL)
C6A6  23          INC   HL           Stepwert
C6A7  66          LD    H,(HL)       nach HL
C6A8  6F          LD    L,A
C6A9  CD AC BD    CALL  BDAC       zu Variablenwert addieren
C6AC  1E 06       LD    E,06       Nr. für "Overflow"
C6AE  D2 94 CA    JP    NC,CA94    Überlauf ? dann Fehler ausg.
C6B1  EB          EX    DE,HL      neuen Variablenwert nach DE
C6B2  E1          POP   HL           Variablenadresse
C6B3  E3          EX    (SP),HL      vom Stack
C6B4  72          LD    (HL),D
C6B5  2B          DEC   HL           neuen Variablenwert
C6B6  73          LD    (HL),E       eintragen
C6B7  E1          POP   HL         Endwert-Zeiger zurück
C6B8  7E          LD    A,(HL)     Endwert lo
C6B9  23          INC   HL         Zeiger auf Endwert hi
C6BA  E5          PUSH  HL         retten
C6BB  66          LD    H,(HL)     Endwert hi
C6BC  6F          LD    L,A        Endwert lo
C6BD  EB          EX    DE,HL      nach DE, FOR-Variablenw. n. HL
C6BE  CD C4 BD    CALL  BDC4       Werte vergleichen
C6C1  E1          POP   HL         Zeiger auf Endwert hi
C6C2  23          INC   HL
C6C3  23          INC   HL           +3= Zeiger auf Stepwert-
C6C4  23          INC   HL           Vorzeichen
C6C5  96          SUB   (HL)       von Vergleichsergebnis abz.
C6C6  C9          RET
```

```
***********************************  Basic-Befehl IF
C6C7  CD FB CE    CALL   CEFB           Ausdruck als Bedingung holen
C6CA  FE A0       CP     A0             Token für GOTO ?
C6CC  28 04       JR     Z,C6D2         dann nicht auf THEN prüfen
C6CE  CD 37 DD    CALL   DD37           Test auf THEN
C6D1  EB                                Token für THEN
C6D2  E5          PUSH   HL             Basic-PC retten
C6D3  CD A3 FD    CALL   FDA3           Vorzeichen von FAC holen
C6D6  E1          POP    HL             Basic-PC zurück
C6D7  CC 9F E8    CALL   Z,E89F         FAC=0 ? dann zugeh. ELSE such.
C6DA  C8          RET    Z              kein ELSE ? dann nächste Zeile
C6DB  CD 51 DD    CALL   DD51           Statementende ?
C6DE  D8          RET    C              dann zurück
C6DF  FE 1E       CP     1E             Zeilennummer ?
C6E1  28 05       JR     Z,C6E8         dann zum GOTO-Befehl
C6E3  FE 1D       CP     1D             Zeilenadresse ?
C6E5  C2 AB DD    JP     NZ,DDAB        nein ? dann folgenden Befehl

***********************************  Basic-Befehl GOTO
C6E8  CD 67 E7    CALL   E767           Zeilenadresse holen
C6EB  EB          EX     DE,HL          als neuen Basic-PC setzen
C6EC  C9          RET

***********************************  Basic-Befehl GOSUB
C6ED  CD 67 E7    CALL   E767           Zeilenadresse holen
C6F0  CD EF E8    CALL   E8EF           bis Statementende überlesen
C6F3  EB          EX     DE,HL          neuer PC in HL, alter in DE
C6F4  0E 00       LD     C,00           Kennzeichen für norm. GOSUB

***********************************  GOSUB-Datensatz auf Stack
                                     IN : DE: einzutragende Adresse
                                           C: Flag für GOSUB-Art
C6F6  E5          PUSH   HL
C6F7  3E 06       LD     A,06           Größe des Eintrags
C6F9  CD B0 F5    CALL   F5B0           Platz auf Basic-Stack reserv.
C6FC  71          LD     (HL),C         GOSUB-Art auf Basic-Stack
C6FD  23          INC    HL
C6FE  73          LD     (HL),E
C6FF  23          INC    HL               einzutragende Adresse
C700  72          LD     (HL),D           auf Basic-Stack
C701  23          INC    HL
C702  EB          EX     DE,HL
C703  CD D2 DD    CALL   DDD2             akt. Zeilenadresse nach DE
C706  EB          EX     DE,HL
C707  73          LD     (HL),E
C708  23          INC    HL               aktuelle Zeilenadresse
C709  72          LD     (HL),D           auf Basic-Stack
C70A  23          INC    HL
C70B  36 06       LD     (HL),06        Eintragsgröße auf Basic-Stack
C70D  E1          POP    HL
C70E  C9          RET

***********************************  Basic-Befehl RETURN
C70F  C0          RET    NZ             Statementende ? sonst Fehler
C710  CD 2E C7    CALL   C72E           GOSUB auf Basic-Stack suchen
C713  CD AC F5    CALL   F5AC           Stackeintrag löschen
C716  4E          LD     C,(HL)         GOSUB-Flag
C717  23          INC    HL
```

```
C718  5E          LD     E,(HL)
C719  23          INC    HL            zwischengespeicherte
C71A  56          LD     D,(HL)        Adresse nach DE
C71B  23          INC    HL
C71C  7E          LD     A,(HL)
C71D  23          INC    HL            alte Zeilenadresse
C71E  66          LD     H,(HL)        nach HL
C71F  6F          LD     L,A
C720  CD CE DD    CALL   DDCE         als neue Zeilenadresse setzen
C723  EB          EX     DE,HL        gespeicherte Adresse nach HL
C724  79          LD     A,C          GOSUB-Flag
C725  FE 01       CP     01           norm. GOSUB ?
C727  D8          RET    C            dann PC=Adresse, zurück
C728  CA A4 C8    JP     Z,C8A4       AFTER-/EVERY- o. ON SQ-GOSUB ?
C72B  C3 B6 C8    JP     C8B6         sonst ON BREAK-GOSUB

*********************************** GOSUB auf Basic-Stack suchen
                                    OUT: HL: Zeiger auf Eintrag
C72E  2A 8B B0    LD     HL,(B08B)    Basic-Stackpointer
C731  2B          DEC    HL           Zeiger auf oberstes Byte
C732  7E          LD     A,(HL)       Länge des obersten Eintrags
C733  F5          PUSH   AF           retten
C734  7D          LD     A,L
C735  96          SUB    (HL)
C736  6F          LD     L,A           Länge abziehen, gibt Zeiger
C737  9F          SBC    A             oberhalb des nächsten
C738  84          ADD    H             Eintrags
C739  67          LD     H,A
C73A  23          INC    HL           DEC HL wieder ausgleichen
C73B  F1          POP    AF           Eintragslänge
C73C  FE 06       CP     06           GOSUB ?
C73E  C8          RET    Z            dann gefunden
C73F  B7          OR     A            nicht Ende des Basic-Stacks ?
C740  20 EF       JR     NZ,C731      dann nächsten Eintrag prüfen
C742  1E 03       LD     E,03         Nr. für "Unexpected RETURN"
C744  C3 94 CA    JP     CA94         Fehler ausgeben

*********************************** Basic-Befehl WHILE
C747  E5          PUSH   HL           Basic-PC retten
C748  CD 18 CA    CALL   CA18         zugehöriges WEND suchen
C74B  E5          PUSH   HL           Zeiger nach WEND-Token
C74C  EB          EX     DE,HL        nach DE, WEND-Zeilenadr. n. HL
C74D  22 2E AC    LD     (AC2E),HL    WEND-Zeilenadresse speichern
C750  CD B8 C7    CALL   C7B8         zugehörige WHILE-Schleife su.
C753  CC AC F5    CALL   Z,F5AC       gef. ? dann vom Stack löschen
C756  3E 07       LD     A,07         Größe des WHILE-Eintrags
C758  CD B0 F5    CALL   F5B0         Platz auf Basic-Stack reserv.
C75B  EB          EX     DE,HL
C75C  CD D2 DD    CALL   DDD2          akt. Zeilenadr. nach DE
C75F  EB          EX     DE,HL
C760  73          LD     (HL),E
C761  23          INC    HL            Zeilenadresse
C762  72          LD     (HL),D        auf Basic-Stack eintragn
C763  23          INC    HL
C764  D1          POP    DE           Zeiger nach WEND-Token
C765  73          LD     (HL),E
C766  23          INC    HL            auf Basic-Stack eintragen
C767  72          LD     (HL),D
```

```
C768  23         INC   HL
C769  EB         EX    DE,HL          Zeiger nach WEND-Token
C76A  E3         EX    (SP),HL        retten, Basic-PC (nach
C76B  EB         EX    DE,HL          WHILE-Token) zurück
C76C  73         LD    (HL),E
C76D  23         INC   HL             Zeiger nach WHILE-Token
C76E  72         LD    (HL),D         auf Basic-Stack
C76F  23         INC   HL
C770  36 07      LD    (HL),07      Eintragsgröße auf Basic-Stack
C772  EB         EX    DE,HL        Basic-PC wieder nach HL
C773  D1         POP   DE           Zeiger nach WEND-Token
C774  18 2A      JR    C7A0         auf Schleifenende prüfen

*********************************** Basic-Befehl WEND
C776  C0         RET   NZ           Statementende ? sonst Fehler
C777  EB         EX    DE,HL        PC (nach WEND-Token) nach DE
C778  CD B8 C7   CALL  C7B8         zugehörige WHILE-Schleife su.
C77B  1E 1E      LD    E,1E         Nr. für "Unexpected WEND"
C77D  C2 94 CA   JP    NZ,CA94      keine Schleife ? dann Fehler
C780  E5         PUSH  HL           Zeiger auf Stack-Eintrag
C781  11 07 00   LD    DE,0007        Eintragsgröße addieren,
C784  19         ADD   HL,DE          gibt Zeiger oberhalb Eintrag
C785  CD AC F5   CALL  F5AC         darüberliegende Schlf. löschen
C788  CD D2 DD   CALL  DDD2         aktuelle Zeilenadresse
C78B  22 2E AC   LD    (AC2E),HL    speichern
C78E  E1         POP   HL           Zeiger auf Stack-Eintrag
C78F  5E         LD    E,(HL)
C790  23         INC   HL             WHILE-Zeilenadresse
C791  56         LD    D,(HL)         laden
C792  23         INC   HL
C793  EB         EX    DE,HL
C794  CD CE DD   CALL  DDCE           als akt. Zeilenadr. setzen
C797  EB         EX    DE,HL
C798  5E         LD    E,(HL)
C799  23         INC   HL             Zeiger nach WEND-Token
C79A  56         LD    D,(HL)         nach DE
C79B  23         INC   HL
C79C  7E         LD    A,(HL)
C79D  23         INC   HL             Zeiger nach WHILE-Token
C79E  66         LD    H,(HL)         als Basic-PC nach HL
C79F  6F         LD    L,A
C7A0  D5         PUSH  DE           Zeiger nach WEND-Token
C7A1  CD FB CE   CALL  CEFB         Ausdruck (WHILE-Bed.) holen
C7A4  E5         PUSH  HL           PC nach Bedingung retten
C7A5  CD A3 FD   CALL  FDA3         Vorzeichen der Bedingung
C7A8  E1         POP   HL           PC nach Bedingung
C7A9  D1         POP   DE           Zeiger nach WEND-Token
C7AA  C0         RET   NZ           Bedingung<>0 (wahr) ?
C7AB  2A 2E AC   LD    HL,(AC2E)    sonst WEND-Zeilenadresse
C7AE  CD CE DD   CALL  DDCE         als akt. Zeilenadr. setzen
C7B1  3E 07      LD    A,07         Größe des Basic-Stack-Eintrags
C7B3  CD A0 F5   CALL  F5A0         Schleife v. Basic-Stack lösch.
C7B6  EB         EX    DE,HL        Zeiger nach WEND-Token als PC
C7B7  C9         RET
```

```
**********************************    offene WHILE-Schleife suchen
                                      IN : DE: Zeiger nach zugeh. WEND
                                           HL: Zeiger auf Eintrag
                                           Z=1, wenn Schleife gefunden
C7B8  2A 8B B0    LD    HL,(B08B)       Basic-Stackpointer
C7BB  2B          DEC   HL              Zeiger auf obersten Eintrag
C7BC  E5          PUSH  HL              retten
C7BD  7D          LD    A,L
C7BE  96          SUB   (HL)              Länge des Eintrags
C7BF  6F          LD    L,A               subtrahieren,
C7C0  9F          SBC   A                 gibt Zeiger auf Eintrag
C7C1  84          ADD   H                 (oberhalb des nächsten)
C7C2  67          LD    H,A
C7C3  23          INC   HL              DEC HL ausgleichen
C7C4  E3          EX    (SP),HL         Zeiger auf Eintrag retten
C7C5  7E          LD    A,(HL)          Länge des Eintrags
C7C6  FE 10       CP    10              Integer-FOR-Schleife ?
C7C8  28 16       JR    Z,C7E0          dann weitersuchen
C7CA  FE 16       CP    16              REAL-FOR-Schleife ?
C7CC  28 12       JR    Z,C7E0          dann weitersuchen
C7CE  FE 07       CP    07              WHILE-Schleife ?
C7D0  20 0C       JR    NZ,C7DE         nein ? dann nicht gefunden
C7D2  2B          DEC   HL
C7D3  2B          DEC   HL
C7D4  2B          DEC   HL                Zeiger auf WEND-Zeiger
C7D5  7E          LD    A,(HL)
C7D6  2B          DEC   HL                Zeiger nach WEND-Token
C7D7  6E          LD    L,(HL)            laden
C7D8  67          LD    H,A
C7D9  CD B8 FF    CALL  FFB8            mit ges. Zeiger vergleichen
C7DC  20 02       JR    NZ,C7E0         ungleich ? dann weitersuchen
C7DE  E1          POP   HL              Zeiger auf Schleifeneintrag
C7DF  C9          RET
C7E0  E1          POP   HL              Zeiger oberhalb nächst. Eintr.
C7E1  18 D8       JR    C7BB            weitersuchen

**********************************    Basic-Befehl ON
C7E3  FE 9C       CP    9C              folgt Token für ERROR ?
C7E5  CA E5 CB    JP    Z,CBE5          dann ON ERROR
C7E8  CD 67 CE    CALL  CE67            Bytewert holen
C7EB  4F          LD    C,A             nach C als Zähler f. Zeilennr.
C7EC  46          LD    B,(HL)          folgendes Zeichen
C7ED  78          LD    A,B
C7EE  FE A0       CP    A0              Token für GOTO ?
C7F0  28 05       JR    Z,C7F7          dann o.k.
C7F2  CD 37 DD    CALL  DD37            Test auf GOSUB
C7F5  9F                                Token für GOSUB
C7F6  2B          DEC   HL              PC wieder auf GOSUB-Token
C7F7  0D          DEC   C               Zähler für Zeilennummern
C7F8  78          LD    A,B             GOTO-/GOSUB-Token
C7F9  CA AB DD    JP    Z,DDAB          Zähler=0 ? dann entspr. Befehl
C7FC  CD 3F DD    CALL  DD3F            sonst nächstes Zeichen
C7FF  CD E1 CE    CALL  CEE1            Zeilennummer holen
C802  FE 2C       CP    2C              folgt Komma ?
C804  28 F1       JR    Z,C7F7          dann weiter behandeln
C806  C9          RET
```

```
************************************  Synchronous Events bearbeiten
C807  AF        XOR   A              Flags für Eventbearbeitung
C808  32 30 AC  LD    (AC30),A       löschen
C80B  CD FB BC  CALL  BCFB           nächstes sync. Event
C80E  30 1D     JR    NC,C82D        laufende Priorität größer ?
C810  47        LD    B,A            Priorität des alten Events
C811  3A 30 AC  LD    A,(AC30)         Flag für "kein DONE SYNC"
C814  E6 7F     AND   7F               löschen
C816  32 30 AC  LD    (AC30),A
C819  C5        PUSH  BC             alte Priorität und
C81A  E5        PUSH  HL             Adr. des Event-Blocks retten
C81B  CD FE BC  CALL  BCFE           Sync. Event ausführen
C81E  E1        POP   HL             Adresse des Event-Blocks und
C81F  C1        POP   BC             alte Priorität zurück
C820  3A 30 AC  LD    A,(AC30)       Bearbeitungs-Flags
C823  17        RLA                  Flag für "kein DONE SYNC" ?
C824  F5        PUSH  AF             Bearbeitungs-Flags retten
C825  78        LD    A,B            alte Priorität
C826  D4 01 BD  CALL  NC,BD01        ggf. KL DONE SYNC, Ev. ausgef.
C829  F1        POP   AF             Bearbeitungs-Flags
C82A  17        RLA                  Flag f. Ende der Sync-Schleife
C82B  30 DE     JR    NC,C80B        nicht gesetzt ? dann weiter
C82D  3A 30 AC  LD    A,(AC30)       Bearbeitungs-Flags
C830  E6 04     AND   04             Flag für "Break ermöglichen" ?
C832  C4 53 C4  CALL  NZ,C453        dann Break-Abbruch ermöglichen
C835  2A 34 AE  LD    HL,(AE34)      PC für auszuführende Routine
C838  3A 30 AC  LD    A,(AC30)       Bearbeitungs-Flags
C83B  E6 03     AND   03             weder Abbruch noch Routine ?
C83D  C8        RET   Z              dann fertig
C83E  1F        RRA                  Flag für Abbruch gesetzt ?
C83F  DA 6B CB  JP    C,CB6B         dann Break ausgeben, Abbruch
C842  23        INC   HL             Zeiger auf Start der Zeile
C843  F1        POP   AF             Aufrufadresse löschen
C844  C3 93 DD  JP    DD93           Routine ausführen

************************************  Break-Event Fortsetzung
C847  22 36 AC  LD    (AC36),HL      Zeiger auf Routinenadresse
C84A  3E 04     LD    A,04           Flag für "Break ermöglichen"
C84C  30 50     JR    NC,C89E        ESC (Break) nicht gedrückt ?
C84E  2A 34 AC  LD    HL,(AC34)      Zeiger auf ON-BREAK-Routine
C851  7C        LD    A,H
C852  B5        OR    L                ON BREAK aktiv ?
C853  C4 D6 DD  CALL  NZ,DDD6        dann Flag für Direkt-Modus h.
C856  3E 41     LD    A,41           Flag f. Abbruch/Schleifenende
C858  30 44     JR    NC,C89E        Direkt-Modus bzw. inaktiv ?
C85A  11 31 AC  LD    DE,AC31        Zeiger auf ON-BREAK-Parameter
C85D  0E 02     LD    C,02           Kennz. für ON BREAK-GOSUB
C85F  18 25     JR    C886           Stackeintr. und Flags setzen

************************************  Zeilenadresse in Event-Block
                                     IN : DE: Adresse des Event-Blocks
C861  D5        PUSH  DE             Adresse retten
C862  CD 37 DD  CALL  DD37           Test auf GOSUB
C865  9F                             Token für GOSUB
C866  CD 67 E7  CALL  E767           Zeilenadresse holen
C869  42        LD    B,D              Zeilenadresse
C86A  4B        LD    C,E              nach BC
C86B  CD 61 DD  CALL  DD61           Spaces, TABs und LFs überlesen
```

```
C86E  D1          POP   DE          Adresse des Event-Blocks
C86F  E5          PUSH  HL          Basic-PC retten
C870  21 0A 00    LD    HL,000A       Offset zu Zeilenadressenfeld
C873  19          ADD   HL,DE         im Parameterfeld addieren
C874  71          LD    (HL),C
C875  23          INC   HL            Zeilenadr. in Parameterfeld
C876  70          LD    (HL),B        des Event-Blocks speichern
C877  E1          POP   HL          Basic-PC zurück
C878  C9          RET

***********************************  Event-Routine f. AFTER/EVERY/SQ
C879  23          INC   HL            Zeiger auf Routinenadresse
C87A  23          INC   HL            im Event-Block +3 ergibt
C87B  23          INC   HL            Adresse des Parameterfelds
C87C  EB          EX    DE,HL       nach DE
C87D  CD D6 DD    CALL  DDD6        Flag für Direkt-Modus holen
C880  3E 40       LD    A,40        Flag für Ende d. Sync-Schleife
C882  30 1A       JR    NC,C89E     Direkt-Modus ?
C884  0E 01       LD    C,01        Flag f. EVERY-/AFTER-/SQ-GOSUB
C886  D5          PUSH  DE          Adresse des Parameter-Feldes
C887  CD F6 C6    CALL  C6F6        GOSUB-Datensatz a. Basic-Stack
C88A  2A 34@AE    LD    HL,(AE34)   Zeiger auf Statementanfang
C88D  EB          EX    DE,HL       nach DE
C88E  E1          POP   HL          Adresse des Parameterfeldes
C88F  70          LD    (HL),B        Priorität des alten Events
C890  23          INC   HL            in Parameterfeld eintragen
C891  73          LD    (HL),E
C892  23          INC   HL            Zeiger auf Statementanfang
C893  72          LD    (HL),D        in Parameterfeld eintragen
C894  23          INC   HL
C895  5E          LD    E,(HL)        Routinenadresse aus
C896  23          INC   HL            Parameterfeld laden, nach DE
C897  56          LD    D,(HL)
C898  EB          EX    DE,HL       und nach HL
C899  22 34 AE    LD    (AE34),HL   und zwischenspeichern
C89C  3E C2       LD    A,C2        Schlf.-Ende/Rout./k. DONE SYNC
C89E  21 30 AC    LD    HL,AC30     Event-Bearbeitungs-Flags
C8A1  B6          OR    (HL)        entsprechende Flags setzen
C8A2  77          LD    (HL),A      und Flags wieder speichern
C8A3  C9          RET

***********************************  RETURN Forts. (AFTER/EVERY/SQ)
C8A4  7E          LD    A,(HL)      Priorität des alten Events
C8A5  23          INC   HL
C8A6  5E          LD    E,(HL)
C8A7  23          INC   HL            alten Basic-PC
C8A8  56          LD    D,(HL)
C8A9  D5          PUSH  DE          retten
C8AA  01 F7 FF    LD    BC,FFF7       Zeiger -7 ergibt
C8AD  09          ADD   HL,BC         Zeiger auf Event-Block
C8AE  CD 01 BD    CALL  BD01        KL DONE SYNC, Event ausgeführt
C8B1  E1          POP   HL          neuer Basic-PC
C8B2  F1          POP   AF          Aufrufadresse löschen
C8B3  C3 74 DD    JP    DD74        zur Interpreterschleife
```

```
************************************  RETURN Forts. (ON BREAK)
C8B6  7E          LD    A,(HL)         Priorität des alten Events
C8B7  2A 36 AC    LD    HL,(AC36)      Zg. auf Rout.-Adr. im Ev.-Bl.
C8BA  01 FC FF    LD    BC,FFFC          -4 gibt Zeiger
C8BD  09          ADD   HL,BC            auf Event-Block
C8BE  CD 01 BD    CALL  BD01           KL DONE SYNC, Event ausgeführt
C8C1  CD 53 C4    CALL  C453           Break-Abbruch wieder ermöglichen
C8C4  2A 32 AC    LD    HL,(AC32)      alten Basic-PC
C8C7  F1          POP   AF             Aufrufadresse löschen
C8C8  C3 74 DD    JP    DD74           zur Interpreterschleife

************************************  Basic-Befehl ON BREAK
C8CB  FE CE       CP    CE             folgt Token für STOP ?
C8CD  11 00 00    LD    DE,0000        Kennz. für ON BREAK inaktiv
C8D0  28 08       JR    Z,C8DA         STOP ? dann abschalten
C8D2  CD 37 DD    CALL  DD37           sonst Test auf GOSUB
C8D5  9F                               Token für GOSUB
C8D6  CD 67 E7    CALL  E767           Zeilenadresse holen
C8D9  2B          DEC   HL             Zeiger auf Zeilenende davor
C8DA  ED 53 34 AC LD    (AC34),DE      Adr. d. ON BREAK-Routine setz.
C8DE  C3 3F DD    JP    DD3F           nächstes Zeichen holen

************************************  Basic-Befehl DI
C8E1  E5          PUSH  HL             Basic-PC retten
C8E2  CD 04 BD    CALL  BD04           KL EVENT DISABLE
C8E5  E1          POP   HL             Basic-PC zurück
C8E6  C9          RET

************************************  Basic-Befehl EI
C8E7  E5          PUSH  HL             Basic-PC retten
C8E8  CD 07 BD    CALL  BD07           KL EVENT ENABLE
C8EB  E1          POP   HL             Basic-PC zurück
C8EC  C9          RET

************************************  Events für Basic initialisieren
C8ED  CD A7 BC    CALL  BCA7           SOUND RESET
C8F0  21 5C AC    LD    HL,AC5C        Zeiger auf Event-Blocks
C8F3  06 04       LD    B,04           Zahl der Event-Blocks
C8F5  E5          PUSH  HL
C8F6  CD EC BC    CALL  BCEC           KL DEL TICKER, Block aushängen
C8F9  E1          POP   HL
C8FA  11 12 00    LD    DE,0012        Länge eines Blocks
C8FD  19          ADD   HL,DE          addieren
C8FE  10 F5       DJNZ  C8F5           weitere Event-Blocks ?
C900  CD 48 BB    CALL  BB48           Break-Taste verriegeln
C903  CD F5 BC    CALL  BCF5           KL SYNC RESET
C906  21 00 00    LD    HL,0000        Kennz. für ON BREAK inaktiv
C909  22 34 AC    LD    (AC34),HL      setzen
C90C  CD 53 C4    CALL  C453           Break-Abbruch ermöglichen
C90F  21 38 AC    LD    HL,AC38        Zeiger auf SQ-Event-Blocks
C912  11 05 03    LD    DE,0305        3 Blocks, 5 Parameter-Bytes
C915  01 00 08    LD    BC,0800        Priorität =8
C918  CD 24 C9    CALL  C924           Event-Blocks generieren
C91B  21 62 AC    LD    HL,AC62        Zg. a. AFTER-/EVERY-Event-Bl.
C91E  11 0B 04    LD    DE,040B        4 Blocks, 11 Parameter-Bytes
C921  01 01 02    LD    BC,0201        Priorität = 2,4,8,16
```

```
***********************************  Event-Block-Gruppe initialisieren
                                     IN : B: Start-Priorität; C: Flag f. wachs. Pr.
                                          D: Zahl d. Blocks; E: Länge d. Param.-Felds
                                          HL: Adresse des 1. Blocks
C924  C5          PUSH   BC
C925  D5          PUSH   DE
C926  0E FD       LD     C,FD          ROM-Konfig., Basic-ROM ein
C928  11 79 C8    LD     DE,C879       Adresse der Event-Routine
C92B  CD EF BC    CALL   BCEF          KL INIT EVENT
C92E  D1          POP    DE            Parameterfeld-Größe
C92F  D5          PUSH   DE
C930  16 00       LD     D,00          Größe hi =0
C932  19          ADD    HL,DE         Größe zu Zeiger addieren
C933  D1          POP    DE
C934  C1          POP    BC
C935  79          LD     A,C           Flag für wachsende Priorität
C936  B7          OR     A
C937  28 03       JR     Z,C93C        nicht gesetzt ?
C939  78          LD     A,B             sonst Priorität
C93A  87          ADD    A               mal 2
C93B  47          LD     B,A
C93C  15          DEC    D             Zähler für Blocks
C93D  20 E5       JR     NZ,C924       weitere Blocks ?
C93F  C9          RET

***********************************  Basic-Befehl ON SQ
C940  CD 37 DD    CALL   DD37          Test auf "("
C943  28                               "("
C944  CD 67 CE    CALL   CE67          Byte-Ausdruck als Kanal holen
C947  F5          PUSH   AF            und retten
C948  CD 5D C9    CALL   C95D          Adresse des Event-Blocks holen
C94B  B7          OR     A             weitere Bits gesetzt ?
C94C  20 1E       JR     NZ,C96C       dann Fehler
C94E  CD 37 DD    CALL   DD37          Test auf ")"
C951  29                               ")"
C952  CD 61 C8    CALL   C861          Zeilenadresse in Event-Block
C955  F1          POP    AF            Kanal
C956  E5          PUSH   HL            Basic-PC retten
C957  EB          EX     DE,HL         Adr. des Event-Blocks nach HL
C958  CD B0 BC    CALL   BCB0          SOUND ARM EVENT
C95B  E1          POP    HL            Basic-PC zurück
C95C  C9          RET

***********************************  Adresse des SQ-Event-Blocks holen
                                     IN : A: Kanal-Byte
                                     OUT: DE: Adresse
C95D  1F          RRA
C95E  11 38 AC    LD     DE,AC38       Adresse für SQ 0
C961  D8          RET    C             entspr. Bit gesetzt ?
C962  1F          RRA
C963  11 44 AC    LD     DE,AC44       Adresse für SQ 1
C966  D8          RET    C             entspr. Bit gesetzt ?
C967  1F          RRA
C968  11 50 AC    LD     DE,AC50       Adresse für SQ 2
C96B  D8          RET    C             entspr. Bit gesetzt ?
C96C  1E 05       LD     E,05          Nr. für "Improper argument"
C96E  C3 94 CA    JP     CA94          Fehler ausgeben
```

```
*********************************** Basic-Befehl AFTER
C971  CD 7C CE     CALL   CE7C        Ticker-Zähler holen
C974  01 00 00     LD     BC,0000     Reload-Zähler =0
C977  18 05        JR     C97E

*********************************** Basic-Befehl EVERY
C979  CD 7C CE     CALL   CE7C        Ticker-Zähler
C97C  42           LD     B,D           Reload-Zähler auf gleichen
C97D  4B           LD     C,E           Wert setzen
C97E  D5           PUSH   DE          Ticker-Zähler
C97F  C5           PUSH   BC          und Reload-Zähler retten
C980  CD 55 DD     CALL   DD55        Test auf Komma
C983  11 00 00     LD     DE,0000     Default-Timer-Nr.
C986  DC 86 CE     CALL   C,CE86      Komma ? dann Timer-Nr. holen
C989  EB           EX     DE,HL       Timer-Nr. nach HL
C98A  CD B1 C9     CALL   C9B1        Adresse des zugeh. Event-Bl.
C98D  E5           PUSH   HL          retten
C98E  01 06 00     LD     BC,0006     Länge des Ticker-Kopfes
C991  09           ADD    HL,BC       zu Event-Block-Adr. addieren
C992  EB           EX     DE,HL       Adresse (ohne Kopf) nach DE
C993  CD 61 C8     CALL   C861        Zeilenadr. in Event-Block
C996  D1           POP    DE          Adresse des Event-Blocks
C997  C1           POP    BC          Reload-Zähler
C998  E3           EX     (SP),HL     PC retten, Ticker-Zähler n. HL
C999  EB           EX     DE,HL       nach DE, Event-Blockadr. n. HL
C99A  CD E9 BC     CALL   BCE9        KL ADD TICKER, Block einhängen
C99D  E1           POP    HL          Basic-PC zurück
C99E  C9           RET

*********************************** Basic-Funktion REMAIN
C99F  CD 8D FE     CALL   FE8D        CINT, FAC nach Integer
C9A2  CD B1 C9     CALL   C9B1        Adr. des zugeh. Event-Blocks
C9A5  CD EC BC     CALL   BCEC        KL DEL TICKER, Block aushängen
C9A8  38 03        JR     C,C9AD      war Block in Ticker Chain ?
C9AA  11 00 00     LD     DE,0000     sonst Null
C9AD  EB           EX     DE,HL       verbleibenden Zähler nach HL
C9AE  C3 0D FF     JP     FF0D        und in FAC eintragen

*********************************** Event-Block-Adresse berechnen
                                      IN : HL: Nummer des Blocks
                                      OUT: HL: Adresse des Blocks
                                                (Block f. EVERY/AFTER)
C9B1  7C           LD     A,H         Blocknr. hi
C9B2  B7           OR     A           <> 0 ?
C9B3  20 B7        JR     NZ,C96C     dann "Improper argument"
C9B5  7D           LD     A,L         Blocknr. lo
C9B6  FE 04        CP     04          >4 ?
C9B8  30 B2        JR     NC,C96C     dann "Improper argument"
C9BA  87           ADD    A
C9BB  87           ADD    A             mal 18, da ein Event-Block
C9BC  87           ADD    A             18 Bytes lang ist
C9BD  85           ADD    L
C9BE  87           ADD    A
C9BF  6F           LD     L,A         gibt Offset f. Block-Tabelle
C9C0  01 5C AC     LD     BC,AC5C     Adr. der Event-Block-Tabelle
C9C3  09           ADD    HL,BC       Offset addieren
C9C4  C9           RET
```

```
***********************************  zugehöriges NEXT suchen
                                     IN : HL: PC nach FOR-Token
                                     OUT: HL: Zeiger nach NEXT-Token
                                          DE: FOR-Zeilenadresse
                                          ($AE36): NEXT-Zeilenadresse
C9C5  EB          EX    DE,HL
C9C6  CD D2 DD    CALL  DDD2              akt. Zeilenadresse nach DE
C9C9  EB          EX    DE,HL
C9CA  2B          DEC   HL              Zeiger auf FOR-Token
C9CB  06 01       LD    B,01            Zähler f. Verschachtelungen
C9CD  0E 1A       LD    C,1A            Nr. für "NEXT missing"
C9CF  CD 23 E9    CALL  E923            nächs. Statem. s., ggf. Fehler
C9D2  E5          PUSH  HL              Suchzeiger vor Statement
C9D3  CD 3F DD    CALL  DD3F            nächstes Zeichen
C9D6  FE B0       CP    B0              Token für NEXT ?
C9D8  28 08       JR    Z,C9E2          dann auswerten
C9DA  E1          POP   HL              Suchzeiger vor Statement
C9DB  FE 9E       CP    9E              Token für FOR ?
C9DD  20 EE       JR    NZ,C9CD         nein ? dann weitersuchen
C9DF  04          INC   B               Verschachtelungstiefe erh.
C9E0  18 EB       JR    C9CD            weiter suchen
C9E2  F1          POP   AF              Suchzeiger löschen
C9E3  EB          EX    DE,HL           akt. Suchzeilenadr. nach HL
C9E4  E5          PUSH  HL              und retten
C9E5  CD D2 DD    CALL  DDD2            aktuelle (FOR-) Zeilenadresse
C9E8  E3          EX    (SP),HL         retten, Suchzeilenadresse
C9E9  CD CE DD    CALL  DDCE            als akt. Zeilenadresse setzen
C9EC  EB          EX    DE,HL           nach DE, Suchzeiger nach HL
C9ED  05          DEC   B               Verschachtelungstiefe
C9EE  28 24       JR    Z,CA14          weitere Verschachtelungen ?
C9F0  CD 3F DD    CALL  DD3F            nächstes Zeichen
C9F3  28 0E       JR    Z,CA03          Statem.-Ende ? dann keine Var.
C9F5  C5          PUSH  BC
C9F6  D5          PUSH  DE
C9F7  CD 86 D6    CALL  D686            (NEXT-)Variable holen/überles.
C9FA  D1          POP   DE
C9FB  C1          POP   BC
C9FC  CD 55 DD    CALL  DD55            folgt Komma ?
C9FF  30 02       JR    NC,CA03         nein ? dann keine weitere Var.
CA01  10 F2       DJNZ  C9F5            weitere Verschachtelungen ?
CA03  2B          DEC   HL              Suchzeiger auf NEXT-Token
CA04  78          LD    A,B             Verschachtelungstiefe
CA05  B7          OR    A               =0 ?
CA06  28 0C       JR    Z,CA14          dann zugehöriges NEXT gefunden
CA08  EB          EX    DE,HL           Suchzeiger nach DE
CA09  CD D2 DD    CALL  DDD2            Suchzeilenadresse holen
CA0C  E3          EX    (SP),HL         retten, FOR-Zeilenadr. nach HL
CA0D  CD CE DD    CALL  DDCE            und als akt. Zeilenadr. setzen
CA10  E1          POP   HL              Suchzeilenadresse
CA11  EB          EX    DE,HL           nach DE, Suchzeiger nach HL
CA12  18 B9       JR    C9CD            weiter suchen
CA14  D1          POP   DE              FOR-Zeilenadresse
CA15  C3 3F DD    JP    DD3F            Zeiger nach NEXT-Token
```

```
*********************************** zugehöriges WEND suchen
                                    IN : HL: PC nach WHILE-Token
                                    OUT: HL: Zeiger nach WEND-Token
                                         DE: WEND-Zeilenadresse
CA18  2B           DEC   HL            Zeiger auf WHILE-Token
CA19  EB           EX    DE,HL
CA1A  CD D2 DD     CALL  DDD2              akt. Zeilenadresse nach DE
CA1D  EB           EX    DE,HL
CA1E  06 00        LD    B,00          Zähler f. Verschachtelungen
CA20  04           INC   B             Zähler erhöhen
CA21  0E 1D        LD    C,1D          Nr. für "WEND missing"
CA23  CD 23 E9     CALL  E923          nächs. Statem. s., ggf. Fehler
CA26  E5           PUSH  HL
CA27  CD 3F DD     CALL  DD3F            nächstes Zeichen
CA2A  E1           POP   HL
CA2B  FE D6        CP    D6            Token für WHILE ?
CA2D  28 F1        JR    Z,CA20        dann Tiefe erh., weiter suchen
CA2F  FE D5        CP    D5            Token für WEND ?
CA31  20 EE        JR    NZ,CA21       nein ? dann weitersuchen
CA33  10 EC        DJNZ  CA21          weitere Verschachtelungen ?
CA35  CD 3F DD     CALL  DD3F          Zeiger auf WEND-Token
CA38  C3 3F DD     JP    DD3F          Zeiger nach WEND-Token

*********************************** Eingabezeile holen
                                    OUT: HL: Zeiger auf Zeile
CA3B  21 A4 AC     LD    HL,ACA4       Zeiger auf Buffer
CA3E  36 00        LD    (HL),00       Kennz. für Buffer leer
CA40  C3 3A BD     JP    BD3A          Zeile holen

*********************************** Buffer ausgeben, Zeile holen
                                    OUT: HL: Zeiger auf Zeile
CA43  21 A4 AC     LD    HL,ACA4       Zeiger auf Buffer
CA46  CD 3A BD     CALL  BD3A          Buffer ausgeben, Zeile holen
CA49  C3 4E C3     JP    C34E          Linefeed ausgeben

*********************************** Zeile von Kassette holen
                                    OUT: HL: Zeigr auf Zeile
                                          CY=0 bei EOF
CA4C  C5           PUSH  BC
CA4D  D5           PUSH  DE
CA4E  21 A4 AC     LD    HL,ACA4       Zeiger auf Buffer
CA51  E5           PUSH  HL            retten
CA52  06 01        LD    B,01          Zeilenlänge=1
CA54  0E 00        LD    C,00          Flag für LF löschen
CA56  CD 80 BC     CALL  BC80          Zeichen von Kassette holen
CA59  CA 6B CB     JP    Z,CB6B        Abbruch ? dann Break ausgeben
CA5C  30 22        JR    NC,CA80       EOF ?
CA5E  77           LD    (HL),A        Zeichen in Buffer speichern
CA5F  FE 0D        CP    0D            CR ?
CA61  28 17        JR    Z,CA7A        dann Ende bzw. LF-CR
CA63  0E 00        LD    C,00          sonst Flag für LF löschen
CA65  FE 0A        CP    0A            LF ?
CA67  20 06        JR    NZ,CA6F       nein ?
CA69  78           LD    A,B           Zeilenlänge
CA6A  3D           DEC   A             =1 ?
CA6B  28 E7        JR    Z,CA54        dann nächstes Zeichen
CA6D  0E FF        LD    C,FF          sonst Flag für LF setzen
CA6F  78           LD    A,B           Zeilenlänge
```

```
CA70  B7          OR    A
CA71  1E 17       LD    E,17          Nr. für "Line too long"
CA73  CA 94 CA    JP    Z,CA94        Länge =$100 ? dann Fehler
CA76  23          INC   HL            Bufferzeiger
CA77  04          INC   B             und Zeilenlänge erhöhen
CA78  18 DC       JR    CA56          nächstes Zeichen
CA7A  79          LD    A,C           LF-Flag
CA7B  B7          OR    A             gesetzt ?
CA7C  20 D8       JR    NZ,CA56       dann nächstes Zeichen
CA7E  77          LD    (HL),A        sonst Null ans Bufferende
CA7F  37          SCF                 C=1 für kein EOF
CA80  E1          POP   HL            Zeiger auf Zeile
CA81  D1          POP   DE
CA82  C1          POP   BC
CA83  C9          RET

**********************************  Fehlernr. und -zeile init.
CA84  AF          XOR   A                Fehlernr. Null

**********************************  Fehlernr. setzen
                                    IN : A: Fehlernr.
CA85  32 AA AD    LD    (ADAA),A         Fehlernr. speichern
CA88  CD D2 DD    CALL  DDD2             akt. Zeilenadresse holen
CA8B  22 A6 AD    LD    (ADA6),HL        und für ERRL speichern
CA8E  C9          RET

**********************************  Basic-Befehl ERROR
CA8F  CD 6D CE    CALL  CE6D             Byte <>0 holen
CA92  C0          RET   NZ               Statementende ? sonst Fehler
CA93  5F          LD    E,A              Byte als Fehlernr. nach E

**********************************  Fehler behandeln
                                    IN : E: Nr. des Fehlers (CPC 464)
                                         A: Nr. des Fehlers
                                                    (CPC 664/6128)
CA94  CD 04 AC    CALL  AC04             User-Vektor
CA97  7B          LD    A,E              Fehlernr.
CA98  CD 85 CA    CALL  CA85             f. ERR speichern, ERRL setzen
CA9B  2A 34 AE    LD    HL,(AE34)        Zeiger auf Statementanfang
CA9E  22 A8 AD    LD    (ADA8),HL        für RESUME speichern
CAA1  CD B0 CB    CALL  CBB0             PC und Zeilenadr. f. CONT sp.
CAA4  31 00 C0    LD    SP,C000          Stackpointer initialisieren
CAA7  2A 32 AE    LD    HL,(AE32)        Basic-SP bei Statementanfang
CAAA  CD AC F5    CALL  F5AC             als Basic-Stackpointer setzen
CAAD  CD B3 FB    CALL  FBB3             Stringdescriptorstack init.
CAB0  CD FD D9    CALL  D9FD             FN-Listenzeiger init.
CAB3  CD DF CA    CALL  CADF             Flag für Direkt-Modus holen
CAB6  2A AF AD    LD    HL,(ADAF)        Adresse der ON ERROR-Routine
CAB9  EB          EX    DE,HL            nach DE
CABA  21 B1 AD    LD    HL,ADB1          Flag f. ON ERROR-Routine aktiv
CABD  30 0C       JR    NC,CACB          Direkt-Modus ? d. Fehler ausg.
CABF  7A          LD    A,D
CAC0  B3          OR    E                  keine ON ERROR-Routine ?
CAC1  28 08       JR    Z,CACB           dann Fehler ausgeben
CAC3  A6          AND   (HL)             ON ERROR-Routine aktiv ?
CAC4  20 05       JR    NZ,CACB          dann Fehler ausgeben
CAC6  35          DEC   (HL)             Flag f. ON ERROR-Routine setz.
CAC7  EB          EX    DE,HL            Adr. der Routine als PC
```

```
CAC8  C3 93 DD     JP     DD93          ON ERROR-Routine ausführen
CACB  36 00        LD     (HL),00       Flag f. ON ERROR-Rout. inaktiv
CACD  3A AA AD     LD     A,(ADAA)      Fehler-Nr.
CAD0  CD 45 CC     CALL   CC45          Adresse d. Fehlerstrings holen
CAD3  2A A6 AD     LD     HL,(ADA6)     Adresse der Fehlerzeile
CAD6  CD CE DD     CALL   DDCE          als akt. Zeilenadresse setzen
CAD9  CD 36 CB     CALL   CB36          Fehlermeldung ausgeben
CADC  C3 64 C0     JP     C064          zur Eingabeschleife

**********************************    Error-Zeilennummer holen
                                      OUT: HL: Zeilennr.
                                           CY=1 für Programm-Modus
                                           CY=0 für Direkt-Modus
CADF  2A A6 AD     LD     HL,(ADA6)       Adresse der Error-Zeile
CAE2  CD D9 DD     CALL   DDD9            Zeilennr. und Flag holen
CAE5  D8           RET    C               Programm-Modus ?
CAE6  21 00 00     LD     HL,0000         sonst Zeilennr. =0
CAE9  C9           RET

**********************************    "Division by zero" ausgeben
CAEA  D5           PUSH   DE
CAEB  E5           PUSH   HL
CAEC  21 13 CD     LD     HL,CD13         Zeiger auf "Division by zero"
CAEF  1E 0B        LD     E,0B            entsprechende Fehlernr.
CAF1  18 07        JR     CAFA            Meldung ggf. ausgeben

**********************************    "Overflow" ausgeben
CAF3  D5           PUSH   DE
CAF4  E5           PUSH   HL
CAF5  21 B9 CC     LD     HL,CCB9         Zeiger auf "Overflow"
CAF8  1E 06        LD     E,06            entsprechende Fehlernr.
CAFA  F5           PUSH   AF
CAFB  E5           PUSH   HL
CAFC  2A AF AD     LD     HL,(ADAF)       Adresse der ON ERROR-Routine
CAFF  7C           LD     A,H               ON ERROR-Routine
CB00  B5           OR     L                 vorhanden ?
CB01  E1           POP    HL
CB02  C2 94 CA     JP     NZ,CA94         dann normale Fehlerbehandlung
CB05  AF           XOR    A               sonst Null
CB06  CD A2 C1     CALL   C1A2            als Streamnr. setzen
CB09  F5           PUSH   AF              alte Streamnr. retten
CB0A  CD 41 C3     CALL   C341            Fehlerstring ausgeben
CB0D  CD 4E C3     CALL   C34E            Linefeed ausgeben
CB10  F1           POP    AF              alte Steamnr.
CB11  CD A2 C1     CALL   C1A2            wieder setzen
CB14  F1           POP    AF
CB15  E1           POP    HL
CB16  D1           POP    DE
CB17  C9           RET

**********************************    "Undefined line xxxxx in yyyyy"
                                      IN : DE: Zeilennr. (xxxxx)
CB18  CD 86 C3     CALL   C386            Bildschirm init.
CB1B  21 23 CB     LD     HL,CB23         Zeiger auf "Undefined line"
CB1E  CD 48 CB     CALL   CB48            mit Zeilennr. ausgeben
CB21  18 1D        JR     CB40            " in" Zeilennr. ausgeben
```

```
**********************************
CB23  55 6E 64 65 66 69 6E 65        Undefine
CB2B  64 20 6C 69 6E 65 20 00        d line .


**********************************  "Break in" Zeilennr. ausgeben
CB33  11 4F CB    LD     DE,CB4F       Zeiger auf "Break"

**********************************  Meldung mit Zeilennr. ausgeben
                                    IN : DE: Zeiger auf Meldung
CB36  CD 9D C1    CALL   C19D          Ein-/Ausgabe init.
CB39  CD 86 C3    CALL   C386          Bildschirm init.
CB3C  EB          EX     DE,HL         Zeiger auf Meldung nach HL
CB3D  CD 41 C3    CALL   C341          Meldung ausgeben
CB40  CD D6 DD    CALL   DDD6          akt. Zeilennr. holen
CB43  D0          RET    NC            Direkt-Modus ? dann fertig
CB44  EB          EX     DE,HL         Zeilennr. nach DE
CB45  21 55 CB    LD     HL,CB55       Zeiger auf " in"
CB48  CD 41 C3    CALL   C341          String ausgeben
CB4B  EB          EX     DE,HL         Zeilennr. nach HL
CB4C  C3 79 EE    JP     EE79          und ausgeben

**********************************
CB4F  42 72 65 61 6B 00                Break.
CB55  20 69 6E 20 00                    in .

**********************************  Basic-Befehl STOP
CB5A  C0          RET    NZ            Statementende ? sonst Fehler
CB5B  E5          PUSH   HL            Basic-PC retten
CB5C  CD 33 CB    CALL   CB33          "Break in" Zeilennr. ausgeben
CB5F  E1          POP    HL            Basic-PC
CB60  CD 93 CB    CALL   CB93          mit Zeilenadresse f. CONT sp.
CB63  18 2B       JR     CB90          zur Eingabeschleife

**********************************  Basic-Befehl END
CB65  C0          RET    NZ            Statementende ? sonst Fehler
CB66  CD 93 CB    CALL   CB93          Zeilenadr. und PC f. CONT sp.
CB69  18 1C       JR     CB87          Files schließen, zur Eingabes.

**********************************  "Break" ausgeben, Abbruch
CB6B  CD 33 CB    CALL   CB33          "Break in" Zeilennr. ausgeben
CB6E  2A 34 AE    LD     HL,(AE34)     Zeiger auf Statementanfang
CB71  CD B0 CB    CALL   CBB0          als PC mit Zeilenadr. f. CONT
CB74  18 1A       JR     CB90          zur Eingabeschleife

**********************************  Programmende behandeln
CB76  CD D6 DD    CALL   DDD6          Flag für Direkt-Modus holen
CB79  30 12       JR     NC,CB8D       bereits Direkt-Modus ?
CB7B  CD AB CB    CALL   CBAB          CONT sperren
CB7E  3A B1 AD    LD     A,(ADB1)      Flag f. ON ERROR-Routine aktiv
CB81  B7          OR     A
CB82  1E 13       LD     E,13          Nr. für "RESUME missing"
CB84  C2 94 CA    JP     NZ,CA94       aktiv ? dann Fehler ausgeben
CB87  CD 98 D2    CALL   D298          CLOSEIN
CB8A  CD A1 D2    CALL   D2A1          CLOSEOUT
CB8D  CD CB DD    CALL   DDCB          Direkt-Modus einschalten
CB90  C3 64 C0    JP     C064          zur Eingabeschleife
```

```
************************************  PC und Zeilenadr. für CONT retten
                                      IN : HL: PC (Statementende !)
CB93  EB          EX     DE,HL
CB94  CD D6 DD    CALL   DDD6            akt. Zeilennr. nach DE
CB97  EB          EX     DE,HL
CB98  D0          RET    NC            Direkt-Modus ? dann fertig
CB99  7E          LD     A,(HL)        aktuelles Zeichen
CB9A  FE 01       CP     01            Token für ":" ?
CB9C  28 0B       JR     Z,CBA9        dann PC setzen
CB9E  23          INC    HL
CB9F  7E          LD     A,(HL)          nächste Zeilenadresse
CBA0  23          INC    HL              =0 (Programmende) ?
CBA1  B6          OR     (HL)
CBA2  28 07       JR     Z,CBAB        dann CONT sperren
CBA4  23          INC    HL            Zeiger auf nächste Zeile
CBA5  CD CE DD    CALL   DDCE          als aktuelle Zeilenadr. setzen
CBA8  23          INC    HL            Zeiger vor Zeilentext
CBA9  18 05       JR     CBB0          mit Zeilenadr. für CONT retten

************************************  CONT sperren
CBAB  21 00 00    LD     HL,0000       Flag für CONT gesperrt
CBAE  18 0C       JR     CBBC          als CONT-PC setzen

************************************  PC und Zeilenadr. für CONT retten
                                      IN : HL: Basic-PC
CBB0  EB          EX     DE,HL         Basic-PC nach DE
CBB1  CD D6 DD    CALL   DDD6          Flag für Direkt-Modus holen
CBB4  D0          RET    NC            Direkt-Modus ?
CBB5  CD D2 DD    CALL   DDD2          Zeilenadresse holen
CBB8  22 AD AD    LD     (ADAD),HL     und für CONT speichern
CBBB  EB          EX     DE,HL         Basic-PC
CBBC  22 AB AD    LD     (ADAB),HL     für CONT speichern
CBBF  C9          RET

************************************  Basic-Befehl CONT
CBC0  C0          RET    NZ            Statementende ? sonst Fehler
CBC1  2A AB AD    LD     HL,(ADAB)     geretteter PC für CONT
CBC4  7C          LD     A,H
CBC5  B5          OR     L               CONT gesperrt ?
CBC6  1E 11       LD     E,11          Nr. für "Cannot continue"
CBC8  CA 94 CA    JP     Z,CA94        gesperrt ? dann Fehler
CBCB  E5          PUSH   HL            Basic-PC retten
CBCC  2A AD AD    LD     HL,(ADAD)     CONT-Zeilenadresse
CBCF  CD CE DD    CALL   DDCE          als akt. Zeilenadresse setzen
CBD2  CD B9 BC    CALL   BCB9          SOUND CONTINUE
CBD5  E1          POP    HL            Basic-PC
CBD6  C3 74 DD    JP     DD74          zur Interpreterschleife

************************************  ON ERROR ausschalten
CBD9  AF          XOR    A               Flag für ON ERROR-Routine
CBDA  32 B1 AD    LD     (ADB1),A        inaktiv
CBDD  11 00 00    LD     DE,0000       Flag f. keine ON ERROR-Routine
CBE0  ED 53 AF AD LD     (ADAF),DE     als Routinenadresse setzen
CBE4  C9          RET

************************************  Basic-Befehl ON ERROR
CBE5  CD 3F DD    CALL   DD3F          ERROR-Token übergehen
CBE8  CD 37 DD    CALL   DD37          Test auf GOTO
```

```
CBEB  A0                              Token für GOTO
CBEC  CD E1 CE    CALL   CEE1         Zeilennummer holen
CBEF  E5          PUSH   HL           Basic-PC retten
CBF0  CD 9A E7    CALL   E79A         Zeile im Programm suchen
CBF3  22 AF AD    LD     (ADAF),HL    als Adr. der ON ERROR-Routine
CBF6  E1          POP    HL           Basic-PC zurück
CBF7  C9          RET

**********************************  Basic-Befehl ON ERROR GOTO 0
CBF8  CD DD CB    CALL   CBDD           ON ERROR ausschalten
CBFB  3A B1 AD    LD     A,(ADB1)       Flag f. ON ERROR-Routine aktiv
CBFE  B7          OR     A
CBFF  C8          RET    Z              nicht gesetzt ? dann o.k.
CC00  C3 A4 CA    JP     CAA4           sonst behandelten Fehler ausg.

**********************************  Basic-Befehl RESUME
CC03  28 14       JR     Z,CC19         Statementende ?
CC05  FE B0       CP     B0             Token für NEXT ?
CC07  28 17       JR     Z,CC20         dann RESUME NEXT
CC09  CD 67 E7    CALL   E767           Zeilenadresse holen
CC0C  CD 4A DD    CALL   DD4A           auf Statementende prüfen
CC0F  D5          PUSH   DE             Zeilenadresse retten
CC10  CD 2B CC    CALL   CC2B           Basic-Zeiger setzen
CC13  E1          POP    HL             Zeilenadresse
CC14  23          INC    HL             Null am Zeilenende übergehen
CC15  F1          POP    AF             Aufrufadresse löschen
CC16  C3 93 DD    JP     DD93           zur Interpreterschleife

**********************************  RESUME ohne Parameter
CC19  CD 2B CC    CALL   CC2B           Basic-Zeiger setzen, PC holen
CC1C  F1          POP    AF             Aufrufadresse löschen
CC1D  C3 74 DD    JP     DD74           zur Interpreterschleife

**********************************  RESUME NEXT
CC20  CD 3F DD    CALL   DD3F           NEXT übergehen
CC23  C0          RET    NZ             Statementende ? sonst Fehler
CC24  CD 2B CC    CALL   CC2B           Basic-Zeiger setzen, PC holen
CC27  23          INC    HL             Zeiger auf nächstes Zeichen
CC28  C3 EF E8    JP     E8EF           nächstes Statement suchen

**********************************  Basic-Zeiger für RESUME setzen
                                    OUT : HL: neuer Basic-PC
CC2B  3A B1 AD    LD     A,(ADB1)       Flag f. ON ERROR-Routine aktiv
CC2E  B7          OR     A
CC2F  1E 14       LD     E,14           Nr. für "Unexpected RESUME"
CC31  CA 94 CA    JP     Z,CA94         inaktiv ? dann Fehler
CC34  AF          XOR    A              Null
CC35  32 AA AD    LD     (ADAA),A       als Fehlernr.
CC38  32 B1 AD    LD     (ADB1),A       ON ERROR-Routine inaktiv
CC3B  2A A6 AD    LD     HL,(ADA6)      Error-Zeilenadresse
CC3E  CD CE DD    CALL   DDCE           als akt. Zeilenadresse setzen
CC41  2A A8 AD    LD     HL,(ADA8)      Error-PC
CC44  C9          RET

**********************************  Adresse des Fehlerstrings holen
                                    IN : A: Fehlernr.
                                    OUT: DE: Adresse
CC45  11 5B CC    LD     DE,CC5B        Adresse der Tabelle
```

```
CC48  FE 1F       CP    1F          Fehlernr. zu groß ?
CC4A  D0          RET   NC          dann "Unknown error"
CC4B  B7          OR    A           Null ?
CC4C  C8          RET   Z           dann "Unknown error"
CC4D  47          LD    B,A         Fehlernr.
CC4E  1A          LD    A,(DE)
CC4F  13          INC   DE            nächsten String suchen
CC50  B7          OR    A             (String-Ende bei Null-Byte)
CC51  20 FB       JR    NZ,CC4E
CC53  05          DEC   B           Zähler für Strings
CC54  20 F8       JR    NZ,CC4E     richtigen erreicht ?
CC56  1A          LD    A,(DE)      1. Zeichen
CC57  B7          OR    A           Null ?
CC58  28 EB       JR    Z,CC45      dann "Unknown error"
CC5A  C9          RET

********************************** Tabelle der Fehlermeldungen
CC5B  55 6E 6B 6E 6F 77 6E 20         0 "Unknown error"
CC63  65 72 72 6F 72 00
CC69  55 6E 65 78 70 65 63 74         1 "Unexpected NEXT"
CC71  65 64 20 4E 45 58 54 00
CC79  53 79 6E 74 61 78 20 65         2 "Syntax error"
CC81  72 72 6F 72 00
CC86  55 6E 65 78 70 65 63 74         3 "Unexpected RETURN"
CC8E  65 64 20 52 45 54 55 52
CC96  4E 00
CC98  44 41 54 41 20 65 78 68         4 "DATA exhausted"
CCA0  61 75 73 74 65 64 00
CCA7  49 6D 70 72 6F 70 65 72         5 "Improper argument"
CCAF  20 61 72 67 75 6D 65 6E
CCB7  74 00
CCB9  4F 76 65 72 66 6C 6F 77         6 "Overflow"
CCC1  00
CCC2  4D 65 6D 6F 72 79 20 66         7 "Memory full"
CCCA  75 6C 6C 00
CCCE  4C 69 6E 65 20 64 6F 65         8 "Line does not exist"
CCD6  73 20 6E 6F 74 20 65 78
CCDE  69 73 74 00
CCE2  53 75 62 73 63 72 69 70         9 "Subscript out of range"
CCEA  74 20 6F 75 74 20 6F 66
CCF2  20 72 61 6E 67 65 00
CCF9  41 72 72 61 79 20 61 6C        10 "Array already dimensioned"
CD01  72 65 61 64 79 20 64 69
CD09  6D 65 6E 73 69 6F 6E 65
CD11  64 00
CD13  44 69 76 69 73 69 6F 6E        11 "Division by zero"
CD1B  20 62 79 20 7A 65 72 6F
CD23  00
CD24  49 6E 76 61 6C 69 64 20        12 "Invalid direct command"
CD2C  64 69 72 65 63 74 20 63
CD34  6F 6D 6D 61 6E 64 00
CD3B  54 79 70 65 20 6D 69 73        13 "Type mismatch"
CD43  6D 61 74 63 68 00
CD49  53 74 72 69 6E 67 20 73        14 "String space full"
CD51  70 61 63 65 20 66 75 6C
CD59  6C 00
CD5B  53 74 72 69 6E 67 20 74        15 "String too long"
CD63  6F 6F 20 6C 6F 6E 67 00
```

```
CD6B  53 74 72 69 6E 67 20 65           16 "String expression too
CD73  78 70 72 65 73 73 69 6F               complex"
CD7B  6E 20 74 6F 6F 20 63 6F
CD83  6D 70 6C 65 78 00
CD89  43 61 6E 6E 6F 74 20 43           17 "Cannot CONTinue"
CD91  4F 4E 54 69 6E 75 65 00
CD99  55 6E 6B 6E 6F 77 6E 20           18 "Unknown user function"
CDA1  75 73 65 72 20 66 75 6E
CDA9  63 74 69 6F 6E 00
CDAF  52 45 53 55 4D 45 20 6D           19 "RESUME missing"
CDB7  69 73 73 69 6E 67 00
CDBE  55 6E 65 78 70 65 63 74           20 "Unexpected RESUME"
CDC6  65 64 20 52 45 53 55 4D
CDCE  45 00
CDD0  44 69 72 65 63 74 20 63           21 "Direct command found"
CDD8  6F 6D 6D 61 6E 64 20 66
CDE0  6F 75 6E 64 00
CDE5  4F 70 65 72 61 6E 64 20           22 "Operand missing"
CDED  6D 69 73 73 69 6E 67 00
CDF5  4C 69 6E 65 20 74 6F 6F           23 "Line too long"
CDFD  20 6C 6F 6E 67 00
CE03  45 4F 46 20 6D 65 74 00           24 "EOF met"
CE0B  46 69 6C 65 20 74 79 70           25 "File type error"
CE13  65 20 65 72 72 6F 72 00
CE1B  4E 45 58 54 20 6D 69 73           26 "NEXT missing"
CE23  73 69 6E 67 00
CE28  46 69 6C 65 20 61 6C 72           27 "File already open"
CE30  65 61 64 79 20 6F 70 65
CE38  6E 00
CE3A  55 6E 6B 6E 6F 77 6E 20           28 "Unknown command"
CE42  63 6F 6D 6D 61 6E 64 00
CE4A  57 45 4E 44 20 6D 69 73           29 "WEND missing"
CE52  73 69 6E 67 00
CE57  55 6E 65 78 70 65 63 74           30 "Unexpected WEND"
CE5F  65 64 20 57 45 4E 44 00

***********************************    Byte-Ausdruck holen
                                       OUT: A: Byte
CE67  CD 86 CE     CALL   CE86            Integer holen, nach DE
CE6A  F5           PUSH   AF              Flag für Statementende
CE6B  18 08        JR     CE75            auf Byte-Wert prüfen

***********************************    Byte-Ausdruck <>0 holen
                                       OUT: A: Byte
CE6D  CD 86 CE     CALL   CE86            Integer holen
CE70  F5           PUSH   AF              Flag für Statementende
CE71  7A           LD     A,D
CE72  B3           OR     E               Ausdruck =0 ?
CE73  28 36        JR     Z,CEAB          dann "Improper argument"
CE75  7A           LD     A,D             Hi-Byte
CE76  B7           OR     A               <>0 ?
CE77  20 32        JR     NZ,CEAB         dann "Improper argument"
CE79  F1           POP    AF              Flag für Statementende
CE7A  7B           LD     A,E             Lo-Byte nach A
CE7B  C9           RET
***********************************    Integer von 0..32767 holen
                                       OUT: DE: Integerwert
CE7C  CD 86 CE     CALL   CE86            Integer holen
```

```
CE7F  F5          PUSH   AF
CE80  7A          LD     A,D          Hi-Byte
CE81  17          RLA                 Vorzeichen negativ ?
CE82  38 27       JR     C,CEAB       dann "Improper argument"
CE84  F1          POP    AF
CE85  C9          RET

**********************************    Integer von -32768..32767 holen
                                      OUT: DE: Integerwert
CE86  CD FB CE    CALL   CEFB            Ausdruck holen
CE89  F5          PUSH   AF
CE8A  EB          EX     DE,HL           Basic-PC nach DE
CE8B  CD 8D FE    CALL   FE8D            CINT, FAC nach Integer in HL
CE8E  EB          EX     DE,HL           Integer nach DE, PC nach HL
CE8F  F1          POP    AF
CE90  C9          RET

**********************************    Integer von -32768..65535 holen
                                      OUT: DE: Integerwert
CE91  CD FB CE    CALL   CEFB            Ausdruck holen
CE94  F5          PUSH   AF
CE95  C5          PUSH   BC
CE96  E5          PUSH   HL
CE97  CD C2 FE    CALL   FEC2            UNT, FAC nach Integer in HL
CE9A  EB          EX     DE,HL           Integerwert nach DE
CE9B  E1          POP    HL
CE9C  C1          POP    BC
CE9D  F1          POP    AF
CE9E  C9          RET

**********************************    String holen, vom Stringstack lö.
                                      OUT: DE: Adrese des Strings
                                           A,B: Stringlänge
CE9F  CD FB CE    CALL   CEFB            Ausdruck holen
CEA2  C3 DA FB    JP     FBDA            String vom Stringstack löschen

**********************************    Stringausdruck holen
CEA5  CD FB CE    CALL   CEFB            Ausdruck holen
CEA8  C3 3C FF    JP     FF3C            auf String prüfen

CEAB  1E 05       LD     E,05            Nr. für "Improper argument"
CEAD  C3 94 CA    JP     CA94            Fehler ausgeben

**********************************    Zeilennummernbereich holen
                                      OUT: BC: Startzeilennummer
                                           DE: Endzeilennummer
                                           CY=1, wenn Komma
                                           Z=1, wenn Statementende/"#"
CEB0  01 01 00    LD     BC,0001         Default-Startzeilennr.
CEB3  11 FF FF    LD     DE,FFFF         Default-Endzeilennr.
CEB6  CD 55 DD    CALL   DD55            folgt Komma ?
CEB9  D4 51 DD    CALL   NC,DD51         sonst Test auf Statementende
CEBC  D8          RET    C               Komma/Statem.-Ende ? dann Def.
CEBD  FE 23       CP     23              "#" ?
CEBF  C8          RET    Z               dann Default
CEC0  FE F5       CP     F5              Token für "-" ?
CEC2  28 0A       JR     Z,CECE          dann keine Startzeilennr.
CEC4  CD E1 CE    CALL   CEE1            Startzeilennr. holen
```

```
CEC7  42          LD    B,D          als Default-Endznr. in DE
CEC8  4B          LD    C,E          als Startzeilennr. nach BC
CEC9  C8          RET   Z          Statementende ? dann Default
CECA  CD 55 DD    CALL  DD55       folgt Komma ?
CECD  D8          RET   C          dann Default
CECE  CD 37 DD    CALL  DD37       Test auf "-"
CED1  F5                           Token für "-"
CED2  11 FF FF    LD    DE,FFFF    Default-Endzeilennr.
CED5  C8          RET   Z          Statementende ? dann Default
CED6  CD 55 DD    CALL  DD55       folgt Komma ?
CED9  D8          RET   C          dann Default
CEDA  CD E1 CE    CALL  CEE1       Endzeilennr. holen
CEDD  C4 55 DD    CALL  NZ,DD55    St.-Ende ? sonst Test a. Komma
CEE0  C9          RET

**********************************  Zeilennummer holen
                                   OUT: DE: Zeilennummer
CEE1  7E          LD    A,(HL)     folgendes Token
CEE2  23          INC   HL
CEE3  5E          LD    E,(HL)       nachfolgende Zeilenadresse
CEE4  23          INC   HL           bzw. Zeilennr. nach DE
CEE5  56          LD    D,(HL)
CEE6  FE 1E       CP    1E         Zeilennummer ?
CEE8  28 0E       JR    Z,CEF8     dann fertig
CEEA  FE 1D       CP    1D         Zeilenadresse ?
CEEC  C2 7B D0    JP    NZ,D07B    nein ? dann "Syntax error"
CEEF  E5          PUSH  HL         Basic-PC retten
CEF0  EB          EX    DE,HL      Zeilenadresse nach HL
CEF1  23          INC   HL
CEF2  23          INC   HL           Zeilenende und Zeilenkopf
CEF3  23          INC   HL           übergehen
CEF4  5E          LD    E,(HL)
CEF5  23          INC   HL           Zeilennummer aus Zeile
CEF6  56          LD    D,(HL)       laden
CEF7  E1          POP   HL         Basic-PC
CEF8  C3 3F DD    JP    DD3F       nächstes Zeichen holen

**********************************  Ausdruck holen
                                   OUT: Ausdruck im FAC
                                        A: Zeichen nach Ausdruck
                                        Z=1 für Statementende
CEFB  C5          PUSH  BC
CEFC  2B          DEC   HL         Zeiger a. Zeichen vor Ausdruck
CEFD  06 00       LD    B,00       Hierarchiecode (HC) bis Ende
CEFF  CD 07 CF    CALL  CF07       Ausdruck bis Ausdrucksende h.
CF02  C1          POP   BC
CF03  2B          DEC   HL         letztes Zeichen des Ausdrucks
CF04  C3 3F DD    JP    DD3F       Zeichen nach Ausdruck holen

**********************************  Teilausdruck holen
                                   (bis zu einem Operator, dessen
                                    Hierarchiecode (HC) <=B ist)
                                   IN : B: Hierachiecode (HC)
CF07  C5          PUSH  BC         End-HC retten
CF08  CD CB CF    CALL  CFCB       1. Operanden holen
CF0B  E5          PUSH  HL         Basic-PC retten

CF0C  E1          POP   HL         Basic-PC
```

```
CF0D  C1          POP   BC          und End-HC zurück
CF0E  7E          LD    A,(HL)      nächstes Zeichen
CF0F  FE EE       CP    EE          kleiner als Token für ">" ?
CF11  D8          RET   C           d. kein Operator, Ausdruckende
CF12  FE FE       CP    FE          größer als Token für XOR ?
CF14  D0          RET   NC          d. kein Operator, Ausdruckende
CF15  FE F4       CP    F4          kleiner als Token für "+" ?
CF17  38 40       JR    C,CF59      dann Vergleichsoperator
CF19  CC 45 FF    CALL  Z,FF45      "+" ? d. Operan. auf Str. prf.
CF1C  20 12       JR    NZ,CF30     kein String oder nicht "+" ?

********************************** Stringverknüpfung "+"
CF1E  C5          PUSH  BC            End-HC retten
CF1F  E5          PUSH  HL            Basic-PC retten
CF20  2A C2 B0    LD    HL,(B0C2)     Zeiger auf 1. Descriptor
CF23  E3          EX    (SP),HL       retten, PC zurück
CF24  CD CB CF    CALL  CFCB          2. Stringoperanden holen
CF27  CD 3C FF    CALL  FF3C          auf String prüfen
CF2A  E3          EX    (SP),HL       PC retten, Zg. auf 1. Descr.
CF2B  CD 63 F8    CALL  F863          Strings verketten
CF2E  18 DC       JR    CF0C          weitere Operanden/Operatoren

********************************** Operator behandeln
CF30  7E          LD    A,(HL)        Operator-Token
CF31  D6 F4       SUB   F4            minus Token für "+"
CF33  87          ADD   A               Nr. in Tabelle mal 4,
CF34  87          ADD   A               da 4 Bytes pro Eintrag
CF35  C6 81       ADD   81
CF37  5F          LD    E,A
CF38  CE CF       ADC   CF              zu Tabellenstart (CF81)
CF3A  93          SUB   E               addieren
CF3B  57          LD    D,A
CF3C  EB          EX    DE,HL         Tabellenzeiger n. HL, PC n. DE
CF3D  78          LD    A,B           End-Hierarchiecode
CF3E  BE          CP    (HL)          mit HC des Operators vergl.
CF3F  EB          EX    DE,HL         PC nach HL, Tab.-Zeiger n. DE
CF40  D0          RET   NC            End-HC >= lfd. HC ? d. fertig
CF41  C5          PUSH  BC            End-HC
CF42  CD 53 FF    CALL  FF53          1. Operanden auf Basic-Stack
CF45  D5          PUSH  DE            Operator-Tabellenadresse
CF46  C5          PUSH  BC            ggf. Vergl.-Nr./Typ d. 1. Opr.
CF47  1A          LD    A,(DE)        laufenden HC aus Tabelle holen
CF48  47          LD    B,A           als End-HC für rekurs. Aufruf
CF49  CD 07 CF    CALL  CF07          2. Operanden rekursiv holen
CF4C  C1          POP   BC            ggf. Vergl.-Nr./Typ d. 1. Opr.
CF4D  E3          EX    (SP),HL       PC retten, Operatortab.-Zeiger
CF4E  23          INC   HL            +1 gibt Aufrufadresse
CF4F  EB          EX    DE,HL         nach DE
CF50  79          LD    A,C           Typ des 1. Operanden
CF51  CD A0 F5    CALL  F5A0          1. Oper. vom Stack, Adr. n. HL
CF54  CD FB FF    CALL  FFFB          Operation ausführen
CF57  18 B3       JR    CF0C          weitere Operanden/Operatoren

********************************** Vergleichsoperator auswerten
CF59  78          LD    A,B           End-HC
CF5A  FE 0A       CP    0A            mit HC für Vergleich vergl.
CF5C  D0          RET   NC            End-HC >= lfd. HC ? d. fertig
CF5D  C5          PUSH  BC            End-HC retten
```

```
CF5E  7E          LD    A,(HL)       Token des Vergleichsoperators
CF5F  D6 ED       SUB   ED           Nr. des Operators von 1..6
CF61  47          LD    B,A          als Maske für <, =, > nach B
CF62  CD 45 FF    CALL  FF45         Stringflag holen
CF65  11 A9 CF    LD    DE,CFA9      Tab.-Adr. f. numer. Vergleich
CF68  20 D8       JR    NZ,CF42      kein String ? dann num. Vergl.
CF6A  E5          PUSH  HL           Basic-PC retten
CF6B  2A C2 B0    LD    HL,(B0C2)    Zeiger auf Descr. d. 1. Oper.
CF6E  E3          EX    (SP),HL      retten, PC zurück
CF6F  C5          PUSH  BC           Vergleichsmaske retten
CF70  06 0A       LD    B,0A         End-HC für Vergleich
CF72  CD 07 CF    CALL  CF07         2. Operanden rekursiv holen
CF75  C1          POP   BC           Vergleichsmaske zurück
CF76  E3          EX    (SP),HL      PC retten, Descr.-Zg. d. 1. O.
CF77  C5          PUSH  BC           Vergleichsmaske retten
CF78  CD 97 F8    CALL  F897         Stringvergleich
CF7B  C1          POP   BC           Vergleichsmaske
CF7C  CD AF CF    CALL  CFAF         Vergl. entspr. Maske auswerten
CF7F  18 8B       JR    CF0C         weitere Operanden/Operatoren

*********************************  Tabelle der Hierarchiecodes und
                                   der Operatorenadressen
CF81  0C
CF82  C3 CC FC    JP    FCCC           +, Addition (numerisch)

CF85  0C
CF86  C3 E1 FC    JP    FCE1           -, Subtraktion

CF89  12
CF8A  C3 F5 FC    JP    FCF5           *, Multiplikation

CF8D  12
CF8E  C3 12 FD    JP    FD12           /, Division

CF91  16
CF92  C3 F4 D4    JP    D4F4           ^, Potenzierung

CF95  10
CF96  C3 37 FD    JP    FD37           \, Integerdivision

CF99  06
CF9A  C3 58 FD    JP    FD58           AND

CF9D  0E
CF9E  C3 49 FD    JP    FD49           MOD

CFA1  04
CFA2  C3 63 FD    JP    FD63           OR

CFA5  02
CFA6  C3 6D FD    JP    FD6D           XOR

CFA9  0A                               numerischer Vergleich

*********************************  numerischer Vergleich
CFAA  C5          PUSH  BC             Vergleichsmaske retten
CFAB  CD 09 FD    CALL  FD09           Vergleich durchf., Ergeb. n. A
CFAE  C1          POP   BC             Vergleichsmaske
```

```
CFAF  C6 01       ADD   01          Erg. $FF f. >, 0 f. =, 1 f. <
CFB1  8F          ADC   A           1 f. >, 2 f. =, 4 f.<
CFB2  A0          AND   B           Erg. in Vergl.-Maske isolieren
CFB3  C6 FF       ADD   FF          CY=0, wenn falsch
CFB5  9F          SBC   A           A=0, wenn falsch, sonst A=$FF
CFB6  C3 05 FF    JP    FF05        Ergebnis in FAC eintragen

*********************************** "-" auswerten
CFB9  2B          DEC   HL          PC vor Ausdruck
CFBA  06 14       LD    B,14        End-HC für Vorzeichenwechsel
CFBC  CD 07 CF    CALL  CF07        Teilausdruck holen
CFBF  C3 89 FD    JP    FD89        Vorzeichenwechsel

*********************************** NOT auswerten
CFC2  2B          DEC   HL          PC vor Ausdruck
CFC3  06 08       LD    B,08        End-HC für NOT
CFC5  CD 07 CF    CALL  CF07        Teilausdruck holen
CFC8  C3 77 FD    JP    FD77        NOT ausführen

*********************************** Einzeloperanden holen
CFCB  CD 3F DD    CALL  DD3F        nächstes Zeichen holen
CFCE  28 1D       JR    Z,CFED      "Operand missing" bei St.-Ende
CFD0  FE 0E       CP    0E          Token für Variable ?
CFD2  38 39       JR    C,D00D      dann auswerten
CFD4  FE 20       CP    20          Token für Konstante ?
CFD6  38 54       JR    C,D02C      dann auswerten
CFD8  FE 22       CP    22          '"' ?
CFDA  CA CB F7    JP    Z,F7CB      dann Stringkonstante holen
CFDD  FE FF       CP    FF          Token für Funktion ?
CFDF  CA 80 D0    JP    Z,D080      dann Funktionsauswertung
CFE2  E5          PUSH  HL          Basic-PC retten
CFE3  21 F2 CF    LD    HL,CFF2     Zeiger auf Tabelle
CFE6  CD 93 FF    CALL  FF93        Adr. entsp. Token aus Tab. h.
CFE9  E3          EX    (SP),HL     Adr. auf Stack, PC zurück
CFEA  C3 3F DD    JP    DD3F        nächstes Z., Routine ausf.

CFED  1E 16       LD    E,16        Nr. für "Operand missing"
CFEF  C3 94 CA    JP    CA94        Fehler ausgeben

*********************************** Tabelle für Operandenauswertung
CFF2  08                            8 Tabelleneinträge
CFF3  78 D0                         D078, "Syntax error"/User-Vek.

CFF5  F5                            Token für "-"
CFF6  B9 CF                         CFB9, Vorzeichenwechsel ausw.

CFF8  F4                            Token für "+"
CFF9  CE CF                         CFCE, Operanden holen

CFFB  28                            "("
CFFC  70 D0                         D070, Ausdruck und ")" holen

CFFE  FE                            Token für NOT
CFFF  C2 CF                         CFC2, NOT auswerten

D001  E3                            Token für ERL
D002  EE D0                         D0EE, ERL auswerten
```

```
D004  E4                                Token für FN
D005  30 D1                             D130, definierte Funktion aw.

D007  AC                                Token für MID$
D008  4B F9                             F94B, MID$-Funktion

D00A  40                                "@"
D00B  FA D0                             D0FA, Variablenadresse holen

*********************************** Variablenwert holen
D00D  CD 90 D6    CALL   D690             Variable holen, Adresse n. DE
D010  30 0B       JR     NC,D01D          existiert Variable nicht ?
D012  FE 03       CP     03               Stringvariable ?
D014  28 0F       JR     Z,D025           dann Descriptor übertragen
D016  E5          PUSH   HL               Basic-PC retten
D017  EB          EX     DE,HL            Variablenadresse nach HL
D018  CD 4B FF    CALL   FF4B             Variable nach FAC
D01B  E1          POP    HL               Basic-PC zurück
D01C  C9          RET
D01D  FE 03       CP     03               kein String ?
D01F  C2 F3 FE    JP     NZ,FEF3          dann FAC löschen
D022  11 2B D0    LD     DE,D02B          Descr.-Zeiger für Leerstring
D025  EB          EX     DE,HL
D026  22 C2 B0    LD     (B0C2),HL          als akt. Descriptoradresse
D029  EB          EX     DE,HL
D02A  C9          RET

D02B  00                                  Stringlänge 0

*********************************** Konstantenwert holen
D02C  D6 0E       SUB    0E               Token in Bereich 0..$11
D02E  FE 0A       CP     0A               Kurz-Konstante <10 ?
D030  38 1D       JR     C,D04F           dann Wert setzen
D032  23          INC    HL               Zeiger auf Konstanten-Wert
D033  FE 0B       CP     0B               Ein-Byte-Konstante ?
D035  28 17       JR     Z,D04E           dann folgendes Byte holen
D037  FE 0F       CP     0F               Zwei-Byte-Konstante ?
D039  38 0E       JR     C,D049           dann auswerten
D03B  FE 11       CP     11               Zeilennr. oder -adresse ?
D03D  38 1A       JR     C,D059           dann auswerten
D03F  20 3A       JR     NZ,D07B          keine REAL-Zahl ? dann Fehler
D041  3E 05       LD     A,05             Typ für REAL, 5 Bytes
D043  CD 4B FF    CALL   FF4B             Wert aus Programm nach FAC
D046  2B          DEC    HL               Basic-PC auf letztes REAL-Byte
D047  18 24       JR     D06D             nächstes Zeichen holen
D049  5E          LD     E,(HL)
D04A  23          INC    HL                 Zwei-Byte-Konstante
D04B  56          LD     D,(HL)             nach DE
D04C  18 04       JR     D052             und nach FAC
D04E  7E          LD     A,(HL)           Ein-Byte-Konstante
D04F  5F          LD     E,A              als Lo-Byte
D050  16 00       LD     D,00             Hi-Byte=0
D052  EB          EX     DE,HL
D053  CD 0D FF    CALL   FF0D               Integerwert in DE nach FAC
D056  EB          EX     DE,HL
D057  18 14       JR     D06D             nächstes Zeichen holen
D059  5E          LD     E,(HL)
D05A  23          INC    HL                 Zeilennr. bzw. -adresse
```

```
D05B  56        LD    D,(HL)       nach DE
D05C  E5        PUSH  HL          Basic-PC retten
D05D  FE 0F     CP    0F          Zeilenadresse ?
D05F  20 07     JR    NZ,D068     nein ? dann Zeilennr. nach FAC
D061  13        INC   DE          Null am Zeilenende übergehen
D062  EB        EX    DE,HL       Zeiger auf Zeile nach HL
D063  23        INC   HL            Zeilenlänge
D064  23        INC   HL            übergehen
D065  5E        LD    E,(HL)
D066  23        INC   HL            Zeilennummer aus Zeile laden
D067  56        LD    D,(HL)        nach DE
D068  EB        EX    DE,HL       Zeilennummer nach HL
D069  CD 60 FE  CALL  FE60        nach REAL wandeln, in FAC
D06C  E1        POP   HL          Basic-PC zurück
D06D  C3 3F DD  JP    DD3F        Zeichen nach Konstante holen

*********************************  Ausdruck und ")" holen
D070  CD FB CE  CALL  CEFB          Ausdruck holen
D073  CD 37 DD  CALL  DD37          Test auf ")"
D076  29                            ")"
D077  C9        RET

D078  CD 0D AC  CALL  AC0D          User-Vektor
D07B  1E 02     LD    E,02          Nr. für "Syntax error"
D07D  C3 94 CA  JP    CA94          Fehler ausgeben

*********************************  Funktionsauswertung
D080  23        INC   HL            Zeiger auf Token
D081  4E        LD    C,(HL)        Funktions-Token laden
D082  CD 3F DD  CALL  DD3F          Zeiger auf nächstes Zeichen
D085  79        LD    A,C           Funktions-Token
D086  FE 40     CP    40            Gruppe 1 (Tokens 0-$1D) ?
D088  38 05     JR    C,D08F        dann Argument holen, ausf.
D08A  FE 49     CP    49            Gruppe 2 (Tokens $40-$48) ?
D08C  DA BB D0  JP    C,D0BB        dann ausführen
D08F  CD 37 DD  CALL  DD37          Test auf "("
D092  28                            "("
D093  79        LD    A,C           Token
D094  87        ADD   A             mal 2, 2 Bytes pro Eintrag
D095  C6 1E     ADD   1E            Offset für Tabelle addieren
D097  4F        LD    C,A           Offset in Tabelle
D098  FE 59     CP    59            größer als max. Tab.-Offset ?
D09A  30 0D     JR    NC,D0A9       dann Fehler
D09C  FE 1D     CP    1D            Gruppe 3 (Tokens $71-$7F) ?
D09E  38 0E     JR    C,D0AE        dann ausführen
D0A0  CD 70 D0  CALL  D070          sonst Argument und ")" holen
D0A3  E5        PUSH  HL            Basic-PC retten
D0A4  CD AE D0  CALL  D0AE          Funktion ausführen
D0A7  E1        POP   HL            Basic-PC zurück
D0A8  C9        RET

D0A9  CD 0A AC  CALL  AC0A          User-Vektor
D0AC  18 CD     JR    D07B          "Syntax error"

*********************************  Funktion anspringen (Gruppen 1/3)
                                   IN : C: Tabellenoffset
D0AE  E5        PUSH  HL            PC retten
D0AF  06 00     LD    B,00          Offset hi =0
```

```
D0B1  21 90 D1       LD      HL,D190        Adresse der Tabelle
D0B4  09             ADD     HL,BC          Offset addieren
D0B5  7E             LD      A,(HL)
D0B6  23             INC     HL               Adresse aus Tabelle
D0B7  66             LD      H,(HL)           laden, nach HL
D0B8  6F             LD      L,A
D0B9  E3             EX      (SP),HL        auf Stack, PC zurück
D0BA  C9             RET                    Funktion anspringen

*********************************        Funktion anspringen (Gruppe 2)
                                         IN : A: Token
                                                  ($40..$48, CPC 464)
                                                  ($40..$49, CPC 664/6128)
D0BB  E5             PUSH    HL             PC retten
D0BC  4F             LD      C,A            Token
D0BD  06 00          LD      B,00           Hi-Byte =0
D0BF  21 4A D0       LD      HL,D04A        Adresse der Tabelle -$80
D0C2  09             ADD     HL,BC            Token mal 2 addieren
D0C3  09             ADD     HL,BC            2 mal $40 gleicht -$80 aus
D0C4  7E             LD      A,(HL)
D0C5  23             INC     HL               Adresse aus Tabelle
D0C6  66             LD      H,(HL)           laden, nach HL
D0C7  6F             LD      L,A
D0C8  E3             EX      (SP),HL        auf Stack, PC zurück
D0C9  C9             RET                    Funktion anspringen

*********************************        Funktionsadressen, Tokens $40-$48
D0CA  17 C4 DC D0 F4 D0 24 FA               EOF, ERR, HIMEM, INKEY$
D0D2  DB D4 84 D5 E5 D0 07 D1               PI, RND, TIME, XPOS
D0DA  0E D1                                 YPOS

*********************************        Basic-Funktion ERR
D0DC  E5             PUSH    HL             Basic-PC retten
D0DD  3A AA AD       LD      A,(ADAA)       Nummer des letzten Fehlers
D0E0  CD 0A FF       CALL    FF0A           in FAC eintragen
D0E3  E1             POP     HL             PC zurück
D0E4  C9             RET

*********************************        Basic-Funktion TIME
D0E5  E5             PUSH    HL             Basic-PC retten
D0E6  CD 0D BD       CALL    BD0D           KL TIME PLEASE, Timer n. DE/HL
D0E9  CD 7C FE       CALL    FE7C           4-Byte-Wert nach FAC
D0EC  E1             POP     HL             PC zurück
D0ED  C9             RET

*********************************        ERL auswerten
D0EE  E5             PUSH    HL             Basic-PC retten
D0EF  CD DF CA       CALL    CADF           Error-Zeilennr. nach HL
D0F2  18 0E          JR      D102           in positive REAL-Zahl wandeln

*********************************        Basic-Funktion HIMEM
D0F4  E5             PUSH    HL             Basic-PC retten
D0F5  2A 7B AE       LD      HL,(AE7B)      HIMEM-Zeiger
D0F8  18 08          JR      D102           in positive REAL-Zahl wandeln

*********************************        Variablenadresse nach FAC ("@")
D0FA  CD 90 D6       CALL    D690           Variable holen, Adr. nach DE
D0FD  D2 AB CE       JP      NC,CEAB        existiert Var. n. ? d. Fehler
```

```
D100  E5          PUSH   HL           Basic-PC retten
D101  EB          EX     DE,HL        Variablenadresse nach HL
D102  CD 60 FE    CALL   FE60         HL in positive REAL-Zahl
D105  E1          POP    HL           Basic-PC zurück
D106  C9          RET

*********************************** Basic-Funktion XPOS
D107  E5          PUSH   HL           Basic-PC retten
D108  CD C6 BB    CALL   BBC6         GRA ASK CURSOR, XPOS nach DE
D10B  EB          EX     DE,HL        X-Koordinate nach HL
D10C  18 04       JR     D112         und in FAC

*********************************** Basic-Funktion YPOS
D10E  E5          PUSH   HL           Basic-PC retten
D10F  CD C6 BB    CALL   BBC6         GRA ASK CURSOR, YPOS nach HL
D112  CD 0D FF    CALL   FF0D         HL in FAC eintragen
D115  E1          POP    HL           Basic-PC zurück
D116  C9          RET

*********************************** Basic-Befehl DEF
D117  CD 37 DD    CALL   DD37         Test auf FN
D11A  E4                              Token für FN
D11B  EB          EX     DE,HL
D11C  CD D6 DD    CALL   DDD6           Flag für Direkt-Modus holen
D11F  EB          EX     DE,HL
D120  1E 0C       LD     E,0C         N. f. "Invalid direct command"
D122  D2 94 CA    JP     NC,CA94      Direkt-Modus ? dann Fehler
D125  CD A2 D6    CALL   D6A2         FN-Eintr. suchen, ggf. anlegen
D128  EB          EX     DE,HL        Zeiger auf Eintrag nach HL
D129  73          LD     (HL),E         aktuellen PC als Zeiger auf
D12A  23          INC    HL             Funktionsdefinition in
D12B  72          LD     (HL),D         FN-Eintrag speichern
D12C  EB          EX     DE,HL        PC wieder nach HL
D12D  C3 EF E8    JP     E8EF         nächstes Statement suchen

*********************************** definierte Funktion auswerten
D130  CD A2 D6    CALL   D6A2         FN-Eintrag suchen
D133  C5          PUSH   BC           Typ des Funktionsresultats und
D134  E5          PUSH   HL           Programmzeiger retten
D135  EB          EX     DE,HL        FN-Eintrags-Zeiger nach HL
D136  5E          LD     E,(HL)
D137  23          INC    HL             Zeiger auf Funktions-
D138  56          LD     D,(HL)         definition nach DE
D139  EB          EX     DE,HL        nach HL
D13A  7C          LD     A,H
D13B  B5          OR     L            Funktion nicht definiert ?
D13C  1E 12       LD     E,12         Nr. f. "Unknown user function"
D13E  CA 94 CA    JP     Z,CA94       ggf. Fehler ausgeben
D141  CD 07 DA    CALL   DA07         neuen Eintrag in FN-Liste
D144  7E          LD     A,(HL)       Zeichen aus Definition
D145  FE 28       CP     28           "(" ?
D147  20 2C       JR     NZ,D175      nein ? dann keine Parameter
D149  CD 3F DD    CALL   DD3F         nächstes Zeichen
D14C  E3          EX     (SP),HL      Def.-PC retten, Aufruf-PC zur.
D14D  CD 37 DD    CALL   DD37         Test auf "("
D150  28                              "("
D151  E3          EX     (SP),HL      Aufruf-PC retten, Def.-PC zur.
D152  CD 4B DA    CALL   DA4B         FN-Var. auf B.-Stack, in Liste
```

```
D155  E3             EX    (SP),HL      Def.-PC retten, Aufruf-PC zur.
D156  D5             PUSH  DE           Zeiger auf Funktions-Variable
D157  CD FB CE       CALL  CEFB         aktuellen Parameter holen
D15A  E3             EX    (SP),HL      Aufr.-PC r., FN-Var.-Zg. n. HL
D15B  78             LD    A,B          Typflag
D15C  CD 66 D6       CALL  D666         Param. an FN-Var. zuweisen
D15F  E1             POP   HL           Aufruf-PC
D160  CD 55 DD       CALL  DD55         folgt Komma ?
D163  30 07          JR    NC,D16C      sonst ausrechnen
D165  E3             EX    (SP),HL      Aufruf-PC retten, Def.-PC zur.
D166  CD 37 DD       CALL  DD37         Test auf Komma
D169  2C                                ","
D16A  18 E6          JR    D152         nächsten akt. Parameter holen
D16C  CD 37 DD       CALL  DD37         Test auf ")"
D16F  29                                ")"
D170  E3             EX    (SP),HL      Def.-PC retten, Aufruf-PC zur.
D171  CD 37 DD       CALL  DD37         Test auf ")"
D174  29                                ")"
D175  CD 27 DA       CALL  DA27         FN-Var.-Liste in FN-Liste
D178  CD 37 DD       CALL  DD37         Test auf "="
D17B  EF                                Token für "="
D17C  CD FB CE       CALL  CEFB         Funktionsresultat berechnen
D17F  C2 7B D0       JP    NZ,D07B      Statementende ? sonst Fehler
D182  CD 30 DA       CALL  DA30         FN-Var.-Liste wieder aushängen
D185  CD 45 FF       CALL  FF45         Typflag des Resultats
D188  CC 49 FB       CALL  Z,FB49       String ? dann ggf. auf S-Stack
D18B  E1             POP   HL           Aufruf-PC
D18C  F1             POP   AF           Typflag des Resultats
D18D  C3 D7 FE       JP    FED7         Res. an Funktionstyp angleich.

********************************** Funktionsadressen, Tokens $71-$7F
D190  BA F8 EA F8 C4 F8 A1 FA           BIN$, DEC$, HEX$, INSTR
D198  3C F9 EE D1 EA D1 76 C2           LEFT$, MAX, MIN, POS
D1A0  43 F9 19 D2 36 FA E9 C4           RIGHT$, ROUND, STRING$, TEST
D1A8  EE C4 AB CE 62 C2                 TESTR, Improper argument, VPOS

********************************** Funktionsadressen, Tokens $00-$1D
D1AE  85 FD 10 FA 3E D5 16 FA           ABS, ASC, ATN, CHR$
D1B6  8D FE 34 D5 EC FE 20 D5           CINT, COS, CREAL, EXP
D1C0  E8 FD 2D FC 09 D4 6D F1           FIX, FRE, INKEY, INP
D1C8  ED FD 23 D4 0A FA 2A D5           INT, JOY, LEN, LOG
D1D0  25 D5 34 F8 58 F1 9F C9           LOG10, LOWER$, PEEK, REMAIN
D1D8  02 FF 2F D5 57 FA 29 D3           SGN, SIN, SPACE$, SQ
D1E0  EF D4 1E F9 39 D5 C2 FE           SQR, STR$, TAN, UNT
D1E8  42 F8 77 FA                       UPPER$, VAL

********************************** Basic-Funktion MIN
D1EA  06 FF          LD    B,FF         Flag für MIN
D1EC  18 02          JR    D1F0

********************************** Basic-Funktion MAX
D1EE  06 01          LD    B,01         Flag für MAX
D1F0  CD FB CE       CALL  CEFB         Ausdruck holen
D1F3  CD 55 DD       CALL  DD55         folgt Komma ?
D1F6  30 1C          JR    NC,D214      nein ? dann fertig
D1F8  CD 53 FF       CALL  FF53         FAC auf Basic-Stack
D1FB  CD FB CE       CALL  CEFB         Ausdruck holen
D1FE  E5             PUSH  HL           Basic-PC retten
```

```
D1FF  79           LD     A,C         Typflag des vorherigen Ausdr.
D200  CD A0 F5     CALL   F5A0        letzten Ausdr. vom Stack
D203  C5           PUSH   BC          Typflag
D204  E5           PUSH   HL          und Zeiger auf letzten Ausdr.
D205  CD 09 FD     CALL   FD09        letzen Ausdruck mit FAC vergl.
D208  E1           POP    HL          Zeiger auf letzten Ausdruck
D209  C1           POP    BC          und Typ zurück
D20A  B7           OR     A           Vergleichsergebnis
D20B  28 04        JR     Z,D211      gleich ? dann nächsten Ausdr.
D20D  B8           CP     B           mit MAX/MIN-Flag vergleichen
D20E  C4 4E FF     CALL   NZ,FF4E     ggf. letz. Ausdr. in FAC holen
D211  E1           POP    HL          Basic-PC zurück
D212  18 DF        JR     D1F3        nächsten Ausdruck holen
D214  CD 37 DD     CALL   DD37        Test auf ")"
D217  29                              ")"
D218  C9           RET

********************************** Basic-Funktion ROUND
D219  CD FB CE     CALL   CEFB        Ausdruck holen
D21C  CD 53 FF     CALL   FF53        FAC auf Basic-Stack
D21F  CD 55 DD     CALL   DD55        folgt Komma ?
D222  11 00 00     LD     DE,0000     Default-Rundungsexponent
D225  DC 86 CE     CALL   C,CE86      ggf. Rundungsexp. holen
D228  CD 37 DD     CALL   DD37        Test auf ")"
D22B  29                              ")"
D22C  E5           PUSH   HL          Basic-PC
D22D  D5           PUSH   DE          Rundungsexponent
D22E  21 27 00     LD     HL,0027     39 (max. Exp.-Betrag)
D231  19           ADD    HL,DE       addieren
D232  11 4F 00     LD     DE,004F     max. Wert
D235  CD B8 FF     CALL   FFB8        Betrag des Exp. >39 ?
D238  D2 AB CE     JP     NC,CEAB     dann "Improper argument"
D23B  D1           POP    DE          Rundungsexponent
D23C  79           LD     A,C         Typ des Arguments
D23D  CD A0 F5     CALL   F5A0        Argument vom Basic-Stack
D240  43           LD     B,E         Rundungsexponent
D241  CD AF FD     CALL   FDAF        Argument runden
D244  E1           POP    HL          Basic-PC zurück
D245  C9           RET

********************************** Basic-Befehl CAT
D246  C0           RET    NZ          Statementende ? sonst Fehler
D247  E5           PUSH   HL          Basic-PC
D248  CD AD D2     CALL   D2AD        Kassette init.
D24B  CD 37 F6     CALL   F637        Output-Buffer reservieren
D24E  CD 9B BC     CALL   BC9B        CAS CATALOG
D251  CD 71 F6     CALL   F671        Output-Buffer freigeben
D254  E1           POP    HL          Basic-PC
D255  C9           RET

********************************** Basic-Befehl OPENOUT
D256  CD 73 D2     CALL   D273        File öffnen, Fehler prüfen

D259  CD 37 F6     CALL   F637        Output-Buffer reservieren
D25C  C3 8C BC     JP     BC8C        CAS OUT OPEN
```

```
*********************************** Basic-Befehl OPENIN
D25F  CD 6A D2    CALL   D26A          Eingabefile öffnen
D262  FE 16       CP     16            Filetyp für Datenfile ?
D264  C8          RET    Z             dann o.k.
D265  1E 19       LD     E,19          Nr. für "File type error"
D267  C3 94 CA    JP     CA94          Fehler augeben

*********************************** Eingabefile öffnen
                                    OUT: A: Filetyp
                                         DE: Startadresse
                                         BC: Länge
D26A  CD 73 D2    CALL   D273          File öffnen

D26D  CD 32 F6    CALL   F632          Input-Buffer reservieren
D270  C3 77 BC    JP     BC77          CAS IN OPEN

*********************************** File öffnen
                                    OUT: A: Filetyp
D273  CD 9F CE    CALL   CE9F          Namen h., Adr. n. DE, Lä. n. B
D276  E3          EX     (SP),HL       PC retten, Aufrufadr. nach HL
D277  EB          EX     DE,HL         nach DE, Adr. des Namens n. HL
D278  CD 85 D2    CALL   D285          File eröffnen
D27B  CA 6B CB    JP     Z,CB6B        Abbruch ? dann "Break"
D27E  E1          POP    HL            PC zurück
D27F  D8          RET    C             kein Fehler ?
D280  1E 1B       LD     E,1B          sonst "File already open"
D282  C3 94 CA    JP     CA94          Fehler ausgeben

*********************************** File öffnen Fortsetzung
                                    IN : DE: Adresse der Open-Routine
                                         HL: Adresse des Namens
                                         B: Länge des Namens
                                    OUT: CY=0, wenn Fehler
                                         CY=0, Z=1, wenn Abbruch
D285  D5          PUSH   DE            Routinenadresse auf Stack
D286  0E 00       LD     C,00          Flag für Meldungen
D288  78          LD     A,B           Länge des Namen
D289  B7          OR     A             =0 ?
D28A  28 08       JR     Z,D294        dann Flag für Meldungen
D28C  7E          LD     A,(HL)        1. Zeichen des Namen
D28D  FE 21       CP     21            "!" ?
D28F  20 03       JR     NZ,D294       nein ? dann Flag für Meldungen
D291  23          INC    HL            sonst "!" übergehen
D292  05          DEC    B             Länge erniedrigen
D293  0D          DEC    C             Flag für keine Meldungen
D294  79          LD     A,C           Flag
D295  C3 6B BC    JP     BC6B          CAS NOISY, danach File öffnen

*********************************** Basic-Befehl CLOSEIN
D298  E5          PUSH   HL            PC retten
D299  CD 7A BC    CALL   BC7A          CAS IN CLOSE
D29C  CD 6D F6    CALL   F66D          Input-Buffer freigeben
D29F  E1          POP    HL            PC
D2A0  C9          RET

*********************************** Basic-Befehl CLOSEOUT
D2A1  E5          PUSH   HL            PC retten
D2A2  CD 8F BC    CALL   BC8F          CAS OUT CLOSE
```

```
D2A5  CA 6B CB     JP    Z,CB6B      Abbruch ? dann "Break"
D2A8  CD 71 F6     CALL  F671        Output-Buffer freigeben
D2AB  E1           POP   HL          PC
D2AC  C9           RET

********************************** Kassette initialisieren
D2AD  C5           PUSH  BC
D2AE  D5           PUSH  DE
D2AF  E5           PUSH  HL
D2B0  CD 7D BC     CALL  BC7D        CAS IN ABANDON
D2B3  CD 6D F6     CALL  F66D        Input-Buffer freigeben
D2B6  CD 92 BC     CALL  BC92        CAS OUT ABANDON
D2B9  CD 71 F6     CALL  F671        Output-Buffer freigeben
D2BC  E1           POP   HL
D2BD  D1           POP   DE
D2BE  C1           POP   BC
D2BF  C9           RET

********************************** Basic-Befehl SOUND
D2C0  CD 67 CE     CALL  CE67        Bytewert holen
D2C3  32 B2 AD     LD    (ADB2),A    als Kanal-Status setzen
D2C6  CD 37 DD     CALL  DD37        Test auf Komma
D2C9  2C                             ","
D2CA  CD FF D3     CALL  D3FF        Tonperiode <4096 holen
D2CD  ED 53 B5 AD  LD    (ADB5),DE   und speichern
D2D1  CD 55 DD     CALL  DD55        Test auf Komma
D2D4  11 14 00     LD    DE,0014     Default-Dauer
D2D7  DC 86 CE     CALL  C,CE86      Komma ? dann Dauer holen
D2DA  ED 53 B9 AD  LD    (ADB9),DE   Dauer speichern
D2DE  01 0C 10     LD    BC,100C     max. und Default-Wert
D2E1  CD 0D D3     CALL  D30D        Lautstärke holen
D2E4  32 B8 AD     LD    (ADB8),A    und speichern
D2E7  0E 00        LD    C,00        Default-Wert
D2E9  CD 0D D3     CALL  D30D        ENV-Folgenr. holen
D2EC  32 B3 AD     LD    (ADB3),A    und speichern
D2EF  CD 0D D3     CALL  D30D        ENT-Folgenr. holen
D2F2  32 B4 AD     LD    (ADB4),A    und speichern
D2F5  06 20        LD    B,20        max. Wert
D2F7  CD 0D D3     CALL  D30D        Geräusch-Folgenr. holen
D2FA  32 B7 AD     LD    (ADB7),A    und speichern
D2FD  CD 4A DD     CALL  DD4A        Test auf Statementende
D300  E5           PUSH  HL          Basic-PC retten
D301  21 B2 AD     LD    HL,ADB2     Zeiger auf SOUND-Parameter
D304  CD AA BC     CALL  BCAA        SOUND QUEUE
D307  E1           POP   HL          Basic-PC zurück
D308  D8           RET   C           in Warteschlange eingehängt ?
D309  F1           POP   AF          sonst Aufrufadresse löschen
D30A  C3 71 DD     JP    DD71        und Befehl nochmal ausführen

********************************** Bytewert für SOUND holen
                                     IN : B: maximaler Wert+1
                                          C: Default-Wert
                                     OUT: A: Bytewert
D30D  CD 55 DD     CALL  DD55        folgt Komma ?
D310  79           LD    A,C         Default-Wert
D311  D0           RET   NC          kein Komma ? dann Default
D312  7E           LD    A,(HL)      nächstes Zeichen
D313  FE 2C        CP    2C          zweites Komma ?
```

```
D315  79          LD    A,C        Default-Wert
D316  C8          RET   Z          2. Komma ? dann Default

********************************* Bytewert kleiner B holen
                                 IN : B: maximaler Wert+1
                                 OUT: A: Bytewert
D317  CD 67 CE    CALL  CE67       Bytewert holen
D31A  B8          CP    B          mit max. Wert vergleichen
D31B  D8          RET   C          o.k. ?
D31C  18 2B       JR    D349       sonst "Improper argument"

********************************* Basic-Befehl RELEASE
D31E  06 08       LD    B,08       max. Wert+1
D320  CD 17 D3    CALL  D317       Kanalbyte<8 holen
D323  E5          PUSH  HL         Basic-PC
D324  CD B3 BC    CALL  BCB3       SOUND RELEASE
D327  E1          POP   HL         PC zurück
D328  C9          RET

********************************* Basic-Funktion SQ
D329  CD 8D FE    CALL  FE8D       CINT, FAC nach Integer
D32C  7D          LD    A,L        Lo-Byte
D32D  B7          OR    A          CY=0
D32E  1F          RRA
D32F  38 06       JR    C,D337     b0 gesetzt ?

D331  1F          RRA
D332  38 03       JR    C,D337     b1 gesetzt ?
D334  1F          RRA              b2 nicht gesetzt ?
D335  30 12       JR    NC,D349    dann "Improper argument"
D337  B4          OR    H          Hi-Byte und restl. Lo-Byte
D338  20 0F       JR    NZ,D349    <>0 ? dann "Improper argument"
D33A  7D          LD    A,L        Kanal-Byte
D33B  CD AD BC    CALL  BCAD       SOUND CHECK
D33E  C3 0A FF    JP    FF0A       Byte in FAC eintragen

********************************* Integer von -128..+127 holen
                                 OUT: DE: Integerwert
D341  CD 86 CE    CALL  CE86       Integerwert holen
D344  7B          LD    A,E        Lo-Byte
D345  87          ADD   A          Vorzeichen ins Carry
D346  9F          SBC   A          $FF, wenn negativ, sonst 0
D347  BA          CP    D          =Hi-Byte ?
D348  C8          RET   Z          dann o.k.
D349  1E 05       LD    E,05       Nr. für "Improper argument"
D34B  C3 94 CA    JP    CA94       Fehler ausgeben

********************************* Basic-Befehl ENV
D34E  CD 6D CE    CALL  CE6D       Byte <>0 als Folgenr. holen
D351  FE 10       CP    10         Byte >=16 ?
D353  30 F4       JR    NC,D349    dann Fehler
D355  F5          PUSH  AF         Folgenr.
D356  11 67 D3    LD    DE,D367    Rout. f. ENV-Param.-Gr. holen
D359  CD D8 D3    CALL  D3D8       max. 5 Parametergruppen holen
D35C  F1          POP   AF         Folgenr.
D35D  E5          PUSH  HL         Basic-PC
D35E  21 BB AD    LD    HL,ADBB    Zeiger auf Parameter-Tabelle
D361  71          LD    (HL),C     Zahl der Gruppen an Tab.-Start
```

```
D362  CD BC BC    CALL   BCBC       SOUND AMPL ENVELOPE
D365  E1          POP    HL         Basic-PC zurück
D366  C9          RET

********************************** Parametergruppe für ENV holen
                                 OUT: C,D,E: geholte Parameter
D367  7E          LD     A,(HL)     Zeichen aus Basic-Text
D368  FE EF       CP     EF         Token für "=" ?
D36A  20 12       JR     NZ,D37E    nein ?
D36C  CD 3F DD    CALL   DD3F       sonst nächstes Zeichen
D36F  06 10       LD     B,10       max Wert+1
D371  CD 17 D3    CALL   D317       Registerwert holen
D374  F6 80       OR     80         Kennz. für "="-Gruppe setzen
D376  4F          LD     C,A        Registerwert nach C
D377  CD 37 DD    CALL   DD37       Test auf Komma
D37A  2C                            ","
D37B  C3 91 CE    JP     CE91       Veränderungsperiode nach DE
D37E  06 80       LD     B,80       max. Wert+1
D380  CD 17 D3    CALL   D317       Schrittanzahl holen
D383  18 40       JR     D3C5       Schrittw. und Pausenzeit holen

********************************** Basic-Befehl ENT
D385  CD 41 D3    CALL   D341       Wert von -128..+127 holen
D388  7A          LD     A,D        Hi-Byte (Vorzeichen)
D389  B7          OR     A
D38A  7B          LD     A,E        Lo-Byte
D38B  28 02       JR     Z,D38F     positiv ?
D38D  2F          CPL                 sonst Betrag
D38E  3C          INC    A            berechnen
D38F  5F          LD     E,A        Wert als Folgenr.
D390  B7          OR     A          =0 ?
D391  28 B6       JR     Z,D349     dann Fehler
D393  FE 10       CP     10         >=16 ?
D395  30 B2       JR     NC,D349    dann Fehler
D397  D5          PUSH   DE         Vorzeichen und Folgenr. retten
D398  11 AE D3    LD     DE,D3AE    Rout. f. ENT-Param.-Gr. holen
D39B  CD D8 D3    CALL   D3D8       max. 5 Parametergruppen holen
D39E  D1          POP    DE         Vorzeichen/Folgenr.
D39F  E5          PUSH   HL         Basic-PC retten
D3A0  21 BB AD    LD     HL,ADBB    Zeiger auf Parameter-Tabelle
D3A3  7A          LD     A,D        Vorzeichen der Folgenr.
D3A4  E6 80       AND    80         b7 isolieren
D3A6  B1          OR     C          als Flag für Wiederholung
D3A7  77          LD     (HL),A     mit Anzahl d. Gruppen an Start
D3A8  7B          LD     A,E        Folgenr.
D3A9  CD BF BC    CALL   BCBF       SOUND TONE ENVELOPE
D3AC  E1          POP    HL         Basic-PC zurück
D3AD  C9          RET

********************************** Parametergruppe für ENT holen
                                 OUT: C,D,E: geholte Parameter
D3AE  7E          LD     A,(HL)     Zeichen aus Basic-Text
D3AF  FE EF       CP     EF         Token für "=" ?
D3B1  20 0D       JR     NZ,D3C0    nein ?
D3B3  CD 3F DD    CALL   DD3F       sonst nächstes Zeichen
D3B6  CD FF D3    CALL   D3FF       Tonperiode <4096 holen
D3B9  7A          LD     A,D        Tonperiode hi
D3BA  C6 F0       ADD    F0         Kennz. für "="-Gruppe setzen
```

```
D3BC  4F          LD    C,A          Tonperiode hi
D3BD  43          LD    B,E          Tonperiode lo
D3BE  18 0E       JR    D3CE         Pausezeit holen
D3C0  06 F0       LD    B,F0         max. Wert+1
D3C2  CD 17 D3    CALL  D317         Schrittanzahl holen
D3C5  4F          LD    C,A          nach C
D3C6  CD 37 DD    CALL  DD37         Test auf Komma
D3C9  2C                             ","
D3CA  CD 41 D3    CALL  D341         Wert von -128..+127 holen
D3CD  43          LD    B,E          als Schrittweite
D3CE  CD 37 DD    CALL  DD37         Test auf Komma
D3D1  2C                             ","
D3D2  CD 67 CE    CALL  CE67         Bytewert holen
D3D5  57          LD    D,A          als Pausenzeit
D3D6  58          LD    E,B          Schrittweite
D3D7  C9          RET

********************************** Parametergruppen f. ENV/ENT holen
                                   IN : DE: Routinenadr. f. 1 Gruppe
                                   OUT: C: Zahl der geholten Gruppen
D3D8  01 00 05    LD    BC,0500        max. Zahl/Zähler f. Gruppen
D3DB  CD 55 DD    CALL  DD55           folgt Komma ?
D3DE  30 1C       JR    NC,D3FC        nein ? dann fertig
D3E0  D5          PUSH  DE
D3E1  C5          PUSH  BC
D3E2  CD FB FF    CALL  FFFB           nächste Gruppe holen
D3E5  79          LD    A,C            1. Parameter-Byte
D3E6  C1          POP   BC             Nr. der Gruppe
D3E7  C5          PUSH  BC
D3E8  E5          PUSH  HL             Basic-PC retten
D3E9  21 BC AD    LD    HL,ADBC        Zeiger auf Tabelle
D3EC  06 00       LD    B,00           Gruppennr. hi =0
D3EE  09          ADD   HL,BC            Gruppennr. 3 mal
D3EF  09          ADD   HL,BC            addieren, da 3 Bytes
D3F0  09          ADD   HL,BC            pro Parameter-Gruppe
D3F1  77          LD    (HL),A
D3F2  23          INC   HL               Parameter-Gruppe
D3F3  73          LD    (HL),E           in Tabelle speichern
D3F4  23          INC   HL
D3F5  72          LD    (HL),D
D3F6  E1          POP   HL             Basic-PC zurück
D3F7  C1          POP   BC             Zähler für Gruppen
D3F8  0C          INC   C              Nr. der Gruppe erhöhen
D3F9  D1          POP   DE
D3FA  10 DF       DJNZ  D3DB           weitere Parameter-Gruppen ?
D3FC  C3 4A DD    JP    DD4A           auf Statementende prüfen

********************************** Integerwert von 0..4095 holen
D3FF  CD 86 CE    CALL  CE86           Integerwert holen
D402  7A          LD    A,D            Hi-Byte
D403  E6 F0       AND   F0             oberes Nibble isolieren
D405  C2 49 D3    JP    NZ,D349        <>0 ? dann Fehler
D408  C9          RET
********************************** Basic-Funktion INKEY
D409  CD 8D FE    CALL  FE8D           CINT, FAC nach Integer
D40C  11 50 00    LD    DE,0050        max. Tastennummer+1
D40F  CD B8 FF    CALL  FFB8           Nr. zu groß ?
D412  30 22       JR    NC,D436        dann "Improper argument"
```

```
D414  7D          LD    A,L          Nr. der Taste
D415  CD 1E BB    CALL  BB1E         KM TEST KEY
D418  21 FF FF    LD    HL,FFFF      -1, Wert für nicht gedrückt
D41B  28 03       JR    Z,D420       Taste nicht gedrückt ?
D41D  69          LD    L,C          CTRL/SHIFT-Flag ins Lo-Byte
D41E  26 00       LD    H,00         Hi-Byte=0
D420  C3 0D FF    JP    FF0D         HL in FAC eintragen

********************************** Basic-Funktion JOY
D423  CD 24 BB    CALL  BB24         KM GET JOYSTICK, nach HL
D426  EB          EX    DE,HL        nach DE
D427  CD 8D FE    CALL  FE8D         CINT, FAC nach Integer nach HL
D42A  7C          LD    A,H
D42B  B5          OR    L            Wert=0 ?
D42C  28 02       JR    Z,D430       dann JOY(0), Wert in D
D42E  53          LD    D,E          sonst JOY(1), Wert in E
D42F  2B          DEC   HL           Argument
D430  7C          LD    A,H
D431  B5          OR    L            Argument =1 ?
D432  7A          LD    A,D          Wert
D433  CA 0A FF    JP    Z,FF0A       ggf. nach FAC
D436  C3 49 D3    JP    D349         sonst "Improper argument"

********************************** Basic-Befehl KEY
D439  FE 8D       CP    8D           folgt Token für "DEF" ?
D43B  28 19       JR    Z,D456       dann KEY DEF
D43D  3E 20       LD    A,20         (??)
D43F  CD 17 D3    CALL  D317         Byte <212 holen (B ist $D4!)
D442  F5          PUSH  AF           Bytewert
D443  CD 37 DD    CALL  DD37         Test auf Komma
D446  2C                             ","
D447  CD 9F CE    CALL  CE9F         String holen, v. St.-Stack lö.
D44A  48          LD    C,B          Stringlänge
D44B  F1          POP   AF           Bytewert
D44C  47          LD    B,A          als Tasten-ASCII-Code
D44D  E5          PUSH  HL           Basic-PC retten
D44E  EB          EX    DE,HL        Zeiger auf String nach HL
D44F  CD 0F BB    CALL  BB0F         KM SET EXPAND
D452  E1          POP   HL           Basic-PC zurück
D453  30 E1       JR    NC,D436      Fehler ?
D455  C9          RET

********************************** Basic-Befehl KEY DEF
D456  CD 3F DD    CALL  DD3F         DEF-Token übergehen
D459  CD 67 CE    CALL  CE67         Byte als Tastennr. holen
D45C  4F          LD    C,A          nach C
D45D  FE 50       CP    50           >= 80 ?
D45F  30 D5       JR    NC,D436      dann "Improper argument"
D461  CD 37 DD    CALL  DD37         Test auf Komma
D464  2C                             ","
D465  06 02       LD    B,02         max. Wert+1
D467  CD 17 D3    CALL  D317         Bytewert holen als Repeatflag
D46A  1F          RRA                Flag ins Carry
D46B  9F          SBC   A            $FF, wenn Flag ges., sonst 0
D46C  47          LD    B,A          Repeat-Flag
D46D  C5          PUSH  BC           und Tastennr. retten
D46E  E5          PUSH  HL           Basic-PC retten
D46F  79          LD    A,C          Tastennr.
```

```
D470  CD 39 BB    CALL   BB39         KM SET REPEAT
D473  E1          POP    HL           Basic-PC zurück
D474  C1          POP    BC           Tastennr. zurück
D475  11 27 BB    LD     DE,BB27      KM SET TRANSLATE
D478  CD 84 D4    CALL   D484         Argument holen, Routine anspr.
D47B  11 2D BB    LD     DE,BB2D      KM SET SHIFT
D47E  CD 84 D4    CALL   D484         Argument holen, Routine anspr.
D481  11 33 BB    LD     DE,BB33      KM SET CONTROL
D484  CD 55 DD    CALL   DD55         folgt Komma ?
D487  D0          RET    NC           nein ? dann zurück
D488  D5          PUSH   DE           Routinenadresse retten
D489  CD 67 CE    CALL   CE67         Bytewert holen
D48C  47          LD     B,A          als ASCII-Wert nach B
D48D  E3          EX     (SP),HL      PC retten, Routinenadr. n. HL
D48E  79          LD     A,C          Nr. der Taste
D48F  CD F8 FF    CALL   FFF8         Routine anspringen
D492  E1          POP    HL           Basic-PC zurück
D493  C9          RET

*********************************** Basic-Befehl SPEED
D494  FE A4       CP     A4             Token für KEY ?
D496  01 3F BB    LD     BC,BB3F        KM SET DELAY
D499  28 10       JR     Z,D4AB         ggf. ausführen
D49B  FE A2       CP     A2             Token für INK ?
D49D  01 3E BC    LD     BC,BC3E        SCR SET FLASHING
D4A0  28 09       JR     Z,D4AB         ggf. ausführen
D4A2  FE D9       CP     D9             Token für WRITE ?
D4A4  28 1D       JR     Z,D4C3         dann SPEED WRITE
D4A6  1E 02       LD     E,02           Nr. für "Syntax error"
D4A8  C3 94 CA    JP     CA94           Fehler ausgeben
D4AB  C5          PUSH   BC             Routinenadresse retten
D4AC  CD 3F DD    CALL   DD3F           nächstes Zeichen
D4AF  CD 6D CE    CALL   CE6D           Byte <>0 holen
D4B2  4F          LD     C,A            als 1. Periodenlänge
D4B3  CD 37 DD    CALL   DD37           Test auf Komma
D4B6  2C                                ","
D4B7  CD 6D CE    CALL   CE6D           Byte <>0 holen
D4BA  5F          LD     E,A            als 2. Periodenlänge
D4BB  51          LD     D,C            1. Periodenlänge
D4BC  C1          POP    BC             Routinenadresse
D4BD  EB          EX     DE,HL          PC n. DE, Periodenlängen n. HL
D4BE  CD F9 FF    CALL   FFF9           Routine anspringen
D4C1  EB          EX     DE,HL          PC wieder nach HL
D4C2  C9          RET
*********************************** Basic-Befehl SPEED WRITE
D4C3  CD 3F DD    CALL   DD3F           WRITE-Token übergehen
D4C6  06 02       LD     B,02           max. Wert+1
D4C8  CD 17 D3    CALL   D317           Byte <2 holen
D4CB  E5          PUSH   HL             Basic-PC retten
D4CC  21 A7 00    LD     HL,00A7        Wert für 2000 Baud
D4CF  3D          DEC    A              SPEED WRITE 1 ?
D4D0  3E 32       LD     A,32           Wert für 2000 Baud
D4D2  28 02       JR     Z,D4D6         dann Werte für 2000 Baud
D4D4  29          ADD    HL,HL            sonst Werte
D4D5  0F          RRCA                    für 1000 Baud
D4D6  CD 68 BC    CALL   BC68           CAS SET SPEED
D4D9  E1          POP    HL             Basic-PC zurück
D4DA  C9          RET
```

```
********************************** Basic-Funktion PI
D4DB  E5          PUSH   HL         Basic-PC retten
D4DC  CD 19 FF    CALL   FF19       FAC-Typ auf REAL setzen
D4DF  CD 1D FF    CALL   FF1D       Zeiger auf FAC holen, nach HL
D4E2  CD 76 BD    CALL   BD76       Wert für PI holen
D4E5  E1          POP    HL         Basic-PC zurück
D4E6  C9          RET

********************************** Basic-Befehl DEG
D4E7  3E FF       LD     A,FF       Flag für DEG
D4E9  18 01       JR     D4EC       setzen

********************************** Basic-Befehl RAD
D4EB  AF          XOR    A          Flag für RAD
D4EC  C3 73 BD    JP     BD73       Flag setzen

********************************** Basic-Funktion SQR
D4EF  01 79 BD    LD     BC,BD79    Routine für SQR
D4F2  18 16       JR     D50A       Funktion ausführen

********************************** Basic-Operator ^
                                  IN : HL: Zeiger auf 1. Operanden
                                        C: Typ des 1. Operanden
                                          (1.Operand = Basis)
                                        2. Operand (Exponent) im FAC
                                  OUT: Potenz im FAC
D4F4  E5          PUSH   HL         Zeiger auf Basis retten
D4F5  C5          PUSH   BC         Typ der Basis retten
D4F6  CD EC FE    CALL   FEEC       CREAL, Exponent nach REAL
D4F9  EB          EX     DE,HL      Zeiger auf Exp. nach DE
D4FA  21 CB AD    LD     HL,ADCB    Zeiger auf Zwischenspeicher
D4FD  CD 3D BD    CALL   BD3D       FAC kopieren
D500  C1          POP    BC         Typ der Basis zurück
D501  E3          EX     (SP),HL    Exp.-Zg. retten, Basiszg. zur.
D502  79          LD     A,C        Typflag der Basis
D503  CD 4B FF    CALL   FF4B       Basis nach FAC
D506  D1          POP    DE         Zeiger auf Exponenten
D507  01 7C BD    LD     BC,BD7C    Routine für Potenzierung

********************************** REAL-Funktion/-Operator ausführen
                                  IN : BC: Routinenadresse
D50A  CD 19 D5    CALL   D519       Funktion ausführen
D50D  D8          RET    C          kein Fehler ?
D50E  CA EA CA    JP     Z,CAEA     ggf. "Division by zero" ausg.
D511  FA F3 CA    JP     M,CAF3     ggf. "Overflow" ausgeben
D514  1E 05       LD     E,05       Nr. für "Improper argument"
D516  C3 94 CA    JP     CA94       Fehler ausgeben

D519  C5          PUSH   BC         Routinenadresse auf Stack
D51A  D5          PUSH   DE
D51B  CD EC FE    CALL   FEEC       CREAL, FAC n. REAL, Zg. n. HL
D51E  D1          POP    DE
D51F  C9          RET               Routine anspringen

********************************** Basic-Funktion EXP
D520  01 85 BD    LD     BC,BD85    Routine für EXP
D523  18 E5       JR     D50A       Funktion ausführen
```

```
*********************************** Basic-Funktion LOG10
D525  01 82 BD      LD     BC,BD82     Routine für LOG10
D528  18 E0         JR     D50A        Funktion ausführen

*********************************** Basic-Funktion LOG
D52A  01 7F BD      LD     BC,BD7F     Routine für LOG
D52D  18 DB         JR     D50A        Funktion ausführen

*********************************** Basic-Funktion SIN
D52F  01 88 BD      LD     BC,BD88     Routine für SIN
D532  18 D6         JR     D50A        Funktion ausführen

*********************************** Basic-Funktion COS
D534  01 8B BD      LD     BC,BD8B     Routine für COS
D537  18 D1         JR     D50A        Funktion ausführen

*********************************** Basic-Funktion TAN
D539  01 8E BD      LD     BC,BD8E     Routine für TAN
D53C  18 CC         JR     D50A        Funktion ausführen

*********************************** Basic-Funktion ATN
D53E  01 91 BD      LD     BC,BD91     Routine für ATN
D541  18 C7         JR     D50A        Funktion ausführen

***********************************
D543  52 61 6E 64 6F 6D 20 6E          Random n
D54B  75 6D 62 65 72 20 73 65          umber se
D553  65 64 20 3F 20 00                ed ? .

*********************************** Basic-Befehl RANDOMIZE
D559  28 06         JR     Z,D561      Statementeende ?
D55B  CD FB CE      CALL   CEFB        sonst Ausdruck holen
D55E  E5            PUSH   HL          Basic-PC retten
D55F  18 1B         JR     D57C        Seed-Wert setzen
D561  E5            PUSH   HL          Basic-PC retten
D562  21 43 D5      LD     HL,D543     Zeiger "Random number seed ? "
D565  CD 41 C3      CALL   C341        String ausgeben
D568  CD 3B CA      CALL   CA3B        Eingabezeile holen
D56B  D2 6B CB      JP     NC,CB6B     Abbruch ? dann "Break"
D56E  CD 4E C3      CALL   C34E        Linefeed ausgeben
D571  CD A3 EC      CALL   ECA3        Eingabe nach binär wandeln
D574  30 EC         JR     NC,D562     Fehler ? dann neue Zeile
D576  CD 61 DD      CALL   DD61        Spaces, TABs und LFs überlesen
D579  B7            OR     A           folgendes Zeichen
D57A  20 E6         JR     NZ,D562     kein Ende ? dann neue Zeile
D57C  CD EC FE      CALL   FEEC        CREAL, FAC nach REAL
D57F  CD 9A BD      CALL   BD9A        RND-Seed-Wert setzen
D582  E1            POP    HL          Basic-PC zurück
D583  C9            RET

*********************************** Basic-Funktion RND
D584  7E            LD     A,(HL)      folgendes Zeichen
D585  FE 28         CP     28          "(" ?
D587  20 1C         JR     NZ,D5A5     nein ? dann RND-Wert holen
D589  CD 3F DD      CALL   DD3F        "(" übergehen
D58C  CD FB CE      CALL   CEFB        Ausdruck holen
D58F  CD 37 DD      CALL   DD37        Test auf ")"
D592  29                               ")"
```

```
D593  E5          PUSH   HL          Basic-PC retten
D594  CD EC FE    CALL   FEEC        CREAL, FAC nach REAL wandeln
D597  CD 70 BD    CALL   BD70        Vorzeichen holen
D59A  20 05       JR     NZ,D5A1     Zahl <>0 ?
D59C  CD A0 BD    CALL   BDA0        sonst letzten RND-Wert holen
D59F  E1          POP    HL          PC zurück
D5A0  C9          RET
D5A1  FC 9A BD    CALL   M,BD9A      negativ ? d. Seed-Wert setzen
D5A4  E1          POP    HL          Basic-PC
D5A5  E5          PUSH   HL
D5A6  CD 16 FF    CALL   FF16        FAC-Typ auf REAL, Zeiger n. HL
D5A9  CD 9D BD    CALL   BD9D        RND-Wert holen
D5AC  E1          POP    HL          PC zurück
D5AD  C9          RET

***********************************  Variablenbereich freigeben
D5AE  CD BE D5    CALL   D5BE          verkettete Listen der Var. lö.
D5B1  2A 83 AE    LD     HL,(AE83)     Zeiger auf Programmende
D5B4  22 85 AE    LD     (AE85),HL     als Variablenstart,
D5B7  22 87 AE    LD     (AE87),HL     Arraystart
D5BA  22 89 AE    LD     (AE89),HL     und Arrayende setzen
D5BD  C9          RET

***********************************  verkettete Listen d. Var. löschen
D5BE  21 D0 AD    LD     HL,ADD0       Tabellenadr. der Listenoffsets
D5C1  3E 36       LD     A,36          27 Offsets für Buchst. und FN
D5C3  CD CB D5    CALL   D5CB          Tabelle löschen

***********************************  verkett. Listen d. Felder löschen
D5C6  21 06 AE    LD     HL,AE06       Tabellenadr. der Listenoffsets
D5C9  3E 06       LD     A,06          3 Offsets für 3 Feldtypen
D5CB  36 00       LD     (HL),00       Tabellenbyte löschen
D5CD  23          INC    HL            Tabellenzeiger
D5CE  3D          DEC    A             Zähler für Bytes
D5CF  20 FA       JR     NZ,D5CB       weitere Bytes ?
D5D1  C9          RET

***********************************  definierte Funktionen löschen
D5D2  21 00 00    LD     HL,0000       Null als
D5D5  22 04 AE    LD     (AE04),HL     1. Offset der VL d. Funktionen
D5D8  C9          RET

***********************************  1. O. der VL der Funktionen holen
D5D9  3E 5B       LD     A,5B          "Z"+1, Nr. in Tabelle

***********************************  1. O. der VL der Variablen holen
                                     IN : A: Anfangsbuchstabe
                                     OUT: BC: Variablenstart-1
                                          HL: Zg. auf 1. Offset der VL
D5DB  2A 85 AE    LD     HL,(AE85)     Start der einfachen Variablen
D5DE  2B          DEC    HL            -1 als Korrektur für Offsets
D5DF  44          LD     B,H             nach BC
D5E0  4D          LD     C,L
D5E1  87          ADD    A             mal 2, da 2 Bytes pro Eintrag
D5E2  C6 4E       ADD    4E
D5E4  6F          LD     L,A             AD4E = ADD0-2*"A" addieren
D5E5  CE AD       ADC    AD              gibt Zeiger in Tabelle
D5E7  95          SUB    L               nach HL
```

```
D5E8  67          LD    H,A
D5E9  C9          RET

*********************************   1. Offset für VL der Felder holen
                                    IN : A: Typflag
                                    OUT: BC: Arraystart-1
                                         HL: Zg. auf 1. Offset der VL
D5EA  2A 87 AE    LD    HL,(AE87)      Start der Arrays
D5ED  2B          DEC   HL             -1, Korrektur für Offset
D5EE  44          LD    B,H              nach BC
D5EF  4D          LD    C,L
D5F0  E6 03       AND   03               Typflag in Offset in Tabelle
D5F2  3D          DEC   A                wandeln, 0 für REAL, 2 für
D5F3  87          ADD   A                Integer, 4 für Strings
D5F4  C6 06       ADD   06
D5F6  6F          LD    L,A              AE06 (Tabellenstart)
D5F7  CE AE       ADC   AE               addieren
D5F9  95          SUB   L                gibt Zeiger in HL
D5FA  67          LD    H,A
D5FB  C9          RET

*********************************   DEFREAL A-Z
D5FC  01 5A 41    LD    BC,415A        "A", "Z" als Grenzen
D5FF  1E 05       LD    E,05           Typflag für REAL

*********************************   DEF-Typflag in Tabelle
                                    IN : A: Typflag
                                         B: Anfangsbuchstabe
                                         C: Endbuchstabe
D601  79          LD    A,C            Endbuchstabe
D602  90          SUB   B              minus Anfangsbuchstabe
D603  38 3D       JR    C,D642         Endbuchst. kleiner ? d. Fehler
D605  E5          PUSH  HL
D606  3C          INC   A              Zahl der Buchstaben
D607  21 CB AD    LD    HL,ADCB        AE0C-"A", Zeiger auf Tabelle
D60A  06 00       LD    B,00           Anfangsbuchstabe hi =0
D60C  09          ADD   HL,BC          Buchstaben addieren
D60D  73          LD    (HL),E         Typflag in Tabelle eintragen
D60E  2B          DEC   HL             Tabellenzeiger
D60F  3D          DEC   A              Zähler
D610  20 FB       JR    NZ,D60D        weitere Buchstaben ?
D612  E1          POP   HL
D613  C9          RET

*********************************   Basic-Befehl DEFSTR
D614  1E 03       LD    E,03           Typflag für String
D616  18 06       JR    D61E

*********************************   Basic-Befehl DEFINT
D618  1E 02       LD    E,02           Typflag für Integer
D61A  18 02       JR    D61E

*********************************   Basic-Befehl DEFREAL
D61C  1E 05       LD    E,05           Typflag für REAL
D61E  7E          LD    A,(HL)         folgendes Zeichen
D61F  CD 71 FF    CALL  FF71           Buchstabe ?
D622  30 1E       JR    NC,D642        nein ? dann Fehler
D624  4F          LD    C,A            als Endbuchstaben
```

```
D625  47        LD    B,A        und Anfangsbuchstaben setzen
D626  CD 3F DD  CALL  DD3F       nächstes Zeichen
D629  FE 2D     CP    2D         "-" ?
D62B  20 0C     JR    NZ,D639    nein ? dann nur ein Buchst.
D62D  CD 3F DD  CALL  DD3F       Zeichen nach "-" holen
D630  CD 71 FF  CALL  FF71       Buchstabe ?
D633  30 0D     JR    NC,D642    nein ? dann Fehler
D635  4F        LD    C,A        als Endbuchstaben
D636  CD 3F DD  CALL  DD3F       Endbuchst. übergehen
D639  CD 01 D6  CALL  D601       DEF-Typflag in Tabelle eintr.
D63C  CD 55 DD  CALL  DD55       folgt Komma ?
D63F  38 DD     JR    C,D61E     dann weitere Buchstaben
D641  C9        RET
D642  1E 02     LD    E,02       Nr. für "Syntax error"
D644  18 06     JR    D64C       Fehler ausgeben

D646  1E 09     LD    E,09       N. f. "Subscript out of range"
D648  18 02     JR    D64C       Fehler ausgeben

D64A  1E 0A     LD    E,0A       "Array already dimensioned"
D64C  C3 94 CA  JP    CA94       Fehler ausgeben

*********************************** LET bzw. RSX-Wort auswerten
D64F  FE F8     CP    F8            Interpretercode = $7C ?
D651  CA A0 F1  JP    Z,F1A0        dann RSX-Wort auswerten

*********************************** Basic-Befehl LET
D654  CD 86 D6  CALL  D686          Variable holen, ggf. anlegen
D657  D5        PUSH  DE            Variablenadresse retten
D658  CD 37 DD  CALL  DD37          Test auf "="
D65B  EF                            Token für "="
D65C  CD FB CE  CALL  CEFB          Ausdruck holen
D65F  78        LD    A,B           Typflag der Variablen
D660  E3        EX    (SP),HL       PC retten, Variablenadr. n. HL
D661  CD 66 D6  CALL  D666          Ausdruck an Variable zuweisen
D664  E1        POP   HL            PC zurück
D665  C9        RET

*********************************** FAC an Variable zuweisen
                                    IN : A: Typflag der Variablen
                                           HL: Variablenadresse
D666  47        LD    B,A           Typflag der Variablen
D667  CD 23 FF  CALL  FF23          Typflag des FAC holen
D66A  B8        CP    B             m. Typflag d. Variablen vergl.
D66B  78        LD    A,B           Typflag der Variablen
D66C  C4 D7 FE  CALL  NZ,FED7       ungleich ? dann FAC angleichen
D66F  CD 45 FF  CALL  FF45          Typ des FAC holen
D672  C2 62 FF  JP    NZ,FF62       kein String ? d. FAC kopieren
D675  E5        PUSH  HL            Zeiger auf Variable retten
D676  CD 59 FB  CALL  FB59          String ggf. kopieren, v. Stack
D679  D1        POP   DE            Zeiger auf Variable
D67A  C3 66 FF  JP    FF66          Descriptor in Var. kopieren

*********************************** Basic-Befehl DIM
D67D  CD B5 D7  CALL  D7B5          eine Variable dimensionieren
D680  CD 55 DD  CALL  DD55          folgt Komma ?
D683  38 F8     JR    C,D67D        dann nächste Variable
D685  C9        RET
```

```
***********************************  Variable holen, ggf. neu anlegen
                                     OUT: DE: Variablenadresse
                                          A,B,C: Typ der Variablen
D686  CD 06 D9    CALL   D906          Variablennamen u. Offset holen
D689  CD DB D7    CALL   D7DB          Adr. berechnen , auf Feld prf.
D68C  38 42       JR     C,D6D0        Offset eingetragen ? d. fertig
D68E  18 28       JR     D6B8          Variable suchen bzw. anlegen

***********************************  Variable holen, nicht anlegen
                                     OUT: DE: Variablenadresse
                                          A,B,C: Typ der Variablen
                                          CY=1, wenn Var. existiert
D690  CD 06 D9    CALL   D906          Variablennamen und Offet holen
D693  CD DB D7    CALL   D7DB          Adr. berechnen , auf Feld prf.
D696  38 38       JR     C,D6D0        Offset eingetragen ? d. fertig
D698  E5          PUSH   HL            Basic-PC zurück
D699  79          LD     A,C           1. Byte des Namens
D69A  CD DB D5    CALL   D5DB          entspr. Adr. d. 1. Offs. d. VL
D69D  CD DE D6    CALL   D6DE          Variable suchen, Adresse holen
D6A0  18 2D       JR     D6CF          Typflag laden, zurück

***********************************  FN-Eintrag suchen, ggf. anlegen
D6A2  CD 06 D9    CALL   D906          FN-Namen und Offset holen
D6A5  38 21       JR     C,D6C8        Offset eingetragen ? d. fertig
D6A7  E5          PUSH   HL            Basic-PC retten
D6A8  CD D9 D5    CALL   D5D9          Adr. d. 1. Offs. der VL holen
D6AB  CD DE D6    CALL   D6DE          FN-Eintrag suchen, Adr. holen
D6AE  D4 3D D7    CALL   NC,D73D       nicht gefunden ? dann anlegen
D6B1  18 1C       JR     D6CF          Typflag laden, zurück

***********************************  einfache Var. holen, ggf. anlegen
D6B3  CD 06 D9    CALL   D906          Variablennamen u. Offset holen
D6B6  38 10       JR     C,D6C8        Offset eingetragen ?
D6B8  E5          PUSH   HL            Basic-PC retten
D6B9  79          LD     A,C           1. Byte des Namens
D6BA  CD DB D5    CALL   D5DB          entspr. Adr. d. 1. Offs. d. VL
D6BD  CD DE D6    CALL   D6DE          Variable suchen, Adresse holen
D6C0  3A C1 B0    LD     A,(B0C1)      Typflag
D6C3  D4 49 D7    CALL   NC,D749       existiert Var. nicht ? d. anl.
D6C6  18 07       JR     D6CF          sonst Typ laden, fertig

***********************************  Adresse aus Offset berechnen
                                     IN : DE: Offset
                                     OUT: DE: Variablenadresse
                                          A,B,C: Typ der Variablen
D6C8  E5          PUSH   HL            Basic-PC retten
D6C9  2A 85 AE    LD     HL,(AE85)     Variablenstart
D6CC  2B          DEC    HL            -1, Korrektur für Offset
D6CD  19          ADD    HL,DE         zu Offset addieren
D6CE  EB          EX     DE,HL         Variablenadresse nach DE
D6CF  E1          POP    HL            Basic-PC zurück
D6D0  3A C1 B0    LD     A,(B0C1)      Typflag
D6D3  47          LD     B,A
D6D4  4F          LD     C,A
D6D5  C9          RET
```

```
***********************************   Variable überlesen, Typ holen
                                      OUT: A,B,C: Typ
D6D6  CD 06 D9     CALL   D906          Variablennamen u. Offset holen
D6D9  CD C1 E8     CALL   E8C1          ggf. Arrayindizes überlesen
D6DC  18 F2        JR     D6D0          Typflag laden

***********************************   Variable suchen
                                      IN : DE: Zeiger a. Variablentoken
                                           HL: Adr. d. 1. Offsets d. VL
                                      OUT: CY=1, wenn gefunden
                                           DE: Adr. des Variablenwerts
                                           CY=0, wenn nicht gefunden
                                             dann HL und DE wie IN
D6DE  D5           PUSH   DE            Zeiger auf Variablentoken
D6DF  EB           EX     DE,HL         Adr. d. 1. Offsets d. VL n. DE
D6E0  2A 2B AE     LD     HL,(AE2B)     Zeiger a. 1. Eint. d. FN-Liste
D6E3  7C           LD     A,H
D6E4  B5           OR     L               kein Eintrag vorhanden ?
D6E5  28 0E        JR     Z,D6F5        dann Var.-Liste durchsuchen
D6E7  D5           PUSH   DE
D6E8  23           INC    HL              +2=Zg. auf VL der Funktions-
D6E9  23           INC    HL              Var. des obersten FN-Aufrufs
D6EA  C5           PUSH   BC
D6EB  01 00 00     LD     BC,0000       Offset=0, da absolute Adresse
D6EE  CD 08 D7     CALL   D708          Var. in FN-Var.-Liste suchen
D6F1  C1           POP    BC
D6F2  38 10        JR     C,D704        gefunden ? dann fertig
D6F4  D1           POP    DE            Adr. d. 1. Offsets d. VL
D6F5  EB           EX     DE,HL         nach HL
D6F6  E5           PUSH   HL            retten
D6F7  CD 08 D7     CALL   D708          Var. in VL suchen
D6FA  38 03        JR     C,D6FF        gefunden ?
D6FC  E1           POP    HL            Adr. d. 1. Offsets d. VL
D6FD  D1           POP    DE            Zeiger auf Variablentoken
D6FE  C9           RET                  CY=0 für nicht gefunden
D6FF  F1           POP    AF
D700  E1           POP    HL            Zeiger auf Variablentoken
D701  C3 6D D7     JP     D76D          Offset d. Var. ins Programm
D704  F1           POP    AF
D705  F1           POP    AF
D706  37           SCF                  CY=1 für gefunden
D707  C9           RET

***********************************   Eintrag in VL suchen
                                      IN : ($AE27): Zeiger ges. Namen
                                           ($B0C1): gesuchter Typ
                                           BC: Basisadresse für Offsets
                                           HL: Zeiger 1. Offset der VL
                                      OUT: CY=1, wenn gefunden
                                           HL: Zg. vor Namen auf Offset
                                           DE: Zeiger nach Typflag
D708  7E           LD     A,(HL)
D709  23           INC    HL              nächsten Offset aus VL laden
D70A  66           LD     H,(HL)
D70B  6F           LD     L,A
D70C  B4           OR     H             Offset =0 ?
D70D  C8           RET    Z             dann Ende der VL, nicht gef.
D70E  09           ADD    HL,BC         Basisadr. zu Offset addieren
```

```
D70F  E5          PUSH  HL         Zeiger auf nächsten Offset
D710  23          INC   HL           +2=Zeiger auf
D711  23          INC   HL           Variablennamen
D712  EB          EX    DE,HL      nach DE
D713  2A 27 AE    LD    HL,(AE27)  Zeiger auf gesuchten Namen
D716  1A          LD    A,(DE)     Byte aus Namen
D717  BE          CP    (HL)       m. gesuchtem Namen vergleichen
D718  20 14       JR    NZ,D72E    ungleich ? d. nächster Eintrag
D71A  23          INC   HL           Zeiger
D71B  13          INC   DE           erhöhen
D71C  17          RLA              Ende des Namens ?
D71D  30 F7       JR    NC,D716    nein ? dann weiter vergleichen
D71F  EB          EX    DE,HL      Zeiger nach Namen nach HL
D720  3A C1 B0    LD    A,(B0C1)   gesuchtes Typflag
D723  3D          DEC   A          auf Variablen-Format
D724  AE          XOR   (HL)       mit Variablen-Typ vergleichen
D725  E6 07       AND   07         FN-Kennz.-Bits löschen
D727  20 05       JR    NZ,D72E    ungleich ? d. nächster Eintrag
D729  EB          EX    DE,HL      Zeiger auf Typ nach DE
D72A  13          INC   DE         Zeiger nach Typ
D72B  E1          POP   HL         Zeiger auf Start des Eintrags
D72C  37          SCF              CY=1 für gefunden
D72D  C9          RET
D72E  E1          POP   HL         Zeiger auf nächsten Offset
D72F  18 D7       JR    D708       weitersuchen

*********************************** Variablennamen überlesen
                                   IN : HL: Zeiger auf Var.-Kopf
                                   OUT: DE: wie HL IN
                                        HL: Zeiger nach Namen
D731  F5          PUSH  AF
D732  54          LD    D,H          Zeiger auf Var.-Kopf
D733  5D          LD    E,L          nach DE retten
D734  23          INC   HL
D735  23          INC   HL           +2=Zeiger auf Name
D736  7E          LD    A,(HL)     Byte aus Namen
D737  23          INC   HL
D738  17          RLA              Name zu Ende ?
D739  30 FB       JR    NC,D736    nein ? dann weiter
D73B  F1          POP   AF
D73C  C9          RET

*********************************** FN-Eintrag neu anlegen
                                   OUT: DE: Adresse des Eintrags
D73D  3E 02       LD    A,02         Größe = 2 Bytes
D73F  CD 49 D7    CALL  D749         Variable neu anlegen
D742  1B          DEC   DE           Zeiger auf Typflag
D743  1A          LD    A,(DE)       Typflag
D744  F6 40       OR    40           Kennz. für FN-Eintrag setzen
D746  12          LD    (DE),A
D747  13          INC   DE           Zeiger wieder zurück
D748  C9          RET

*********************************** einfache Variable neu anlegen
                                   IN : A: Typflag
                                        BC: Start der Variablen-1
                                        DE: Zeiger auf Var.-Token
                                        HL: Adr. d. 1. Offsets d. VL
```

```
                                         OUT: DE: Adresse d. Var.-Eintrags
                                              CY=1, da Variable existiert
D749  D5          PUSH   DE
D74A  E5          PUSH   HL
D74B  C5          PUSH   BC
D74C  F5          PUSH   AF               ben. Platz für Variablenwert
D74D  CD 77 D7    CALL   D777             Namenlänge holen, Platz ber.
D750  F5          PUSH   AF               Namenlänge
D751  2A 87 AE    LD     HL,(AE87)        Zeiger auf Arraystart
D754  EB          EX     DE,HL            als Einfügestelle f. neue Var.
D755  CD F8 F5    CALL   F5F8             Platz schaffen, Arrays versch.
D758  CD 3A F5    CALL   F53A             Array-Zeiger entspr. korrig.
D75B  F1          POP    AF               Länge des Namen
D75C  CD 8A D7    CALL   D78A             Namen und Typ übertragen
D75F  F1          POP    AF               Typ/Länge des Variablenwertes
D760  2B          DEC    HL
D761  36 00       LD     (HL),00          Variablenwert löschen
D763  3D          DEC    A
D764  20 FA       JR     NZ,D760          weitere Bytes ?
D766  C1          POP    BC               Start der Variablen-1
D767  E3          EX     (SP),HL          Var.-Adr. retten, VL-Adr. zur.
D768  CD A5 D7    CALL   D7A5             Variable in VL eintragen
D76B  D1          POP    DE               Adresse des Variablenwertes
D76C  E1          POP    HL               Zeiger auf Variablen-Token

***********************************       Variablen-Offset ins Programm sp.
                                          IN : HL: Zeiger auf Var.-Token
                                               DE: Adresse des Var.-Eintr.
                                               BC: Basisadr. für Offset
                                          OUT: CY=1, da Variable existiert
D76D  23          INC    HL                  Zeiger auf Offset
D76E  7B          LD     A,E                   Adr. der Variable/des Feldes
D76F  91          SUB    C                     minus Basisadresse für
D770  77          LD     (HL),A                Offsets gibt Offset für
D771  23          INC    HL                    Variable/Feld, ins Programm
D772  7A          LD     A,D                   eintragen, um schnelles
D773  98          SBC    B                     Auffinden zu ermöglichen
D774  77          LD     (HL),A
D775  37          SCF                        CY=1, weil Variable existiert
D776  C9          RET

***********************************       Namenlänge holen, Platz berechnen
                                          IN : A: Typflag
                                          OUT: A: Länge des Namens
                                               BC: benötigter Platz
D777  C6 03       ADD    03               1 B. f. Typ, 2 B. f. VL-Offs.
D779  4F          LD     C,A              ben. Platz nach C
D77A  2A 27 AE    LD     HL,(AE27)        Adresse des Namens
D77D  06 00       LD     B,00             Zähler für Namenlänge
D77F  0C          INC    C                ben. Platz erhöhen
D780  04          INC    B                Namenlänge erhöhen
D781  7E          LD     A,(HL)           Byte aus Namen
D782  23          INC    HL
D783  17          RLA                     Ende des Namens ?
D784  30 F9       JR     NC,D77F          nein ? dann weiter
D786  78          LD     A,B              Länge des Namens
D787  06 00       LD     B,00             benötigter Platz hi =0
D789  C9          RET
```

```
**********************************   Namen und Typ übertragen
                                     IN : A: Länge des Namens
                                          DE: Zeiger auf Eintrag
                                          BC: Länge des Eintrags
                                     OUT: DE wie IN
                                          HL: Zg. auf Ende d. Eintrags
                                          BC: Zeiger nach Typflag
D78A  62        LD    H,D                 Zeiger auf Eintrag
D78B  6B        LD    L,E                 nach HL
D78C  09        ADD   HL,BC             Länge addieren, Zg. auf Ende
D78D  4F        LD    C,A                 Länge des Namens
D78E  06 00     LD    B,00                nach BC
D790  E5        PUSH  HL                Zeiger auf Ende des Eintrags
D791  D5        PUSH  DE                Zeiger auf Eintrag
D792  13        INC   DE
D793  13        INC   DE                  +2=Zeiger auf Platz f. Namen
D794  2A 27 AE  LD    HL,(AE27)         Zeiger auf Namen
D797  CD F2 FF  CALL  FFF2              Namen kopieren
D79A  3A C1 B0  LD    A,(B0C1)          Typ
D79D  3D        DEC   A                 auf Variablenformat
D79E  12        LD    (DE),A            und übertragen
D79F  13        INC   DE                Zeiger nach Typ (auf Wert)
D7A0  42        LD    B,D                 nach BC
D7A1  4B        LD    C,E
D7A2  D1        POP   DE
D7A3  E1        POP   HL
D7A4  C9        RET

**********************************   Eintrag in VL einhängen
                                     IN : HL: Zeiger a. 1. Offs. d. VL
                                          DE: Zeiger auf Eintrag
                                          BC: Basisadresse für Offsets
                                     OUT: DE: Zeiger auf Nameneintrag
                                          BC: wie IN
D7A5  7E        LD    A,(HL)              bisherigen 1. Offset der VL
D7A6  12        LD    (DE),A              in neuen Eintrag speichern,
D7A7  7B        LD    A,E                 Zeiger auf diesen Eintrag
D7A8  91        SUB   C                   minus Basisadresse
D7A9  77        LD    (HL),A              als neuen 1. Offset setzen
D7AA  23        INC   HL                    für Lo-Byte
D7AB  7E        LD    A,(HL)
D7AC  F5        PUSH  AF
D7AD  7A        LD    A,D
D7AE  98        SBC   B
D7AF  77        LD    (HL),A                und analog für Hi-Byte
D7B0  F1        POP   AF
D7B1  13        INC   DE
D7B2  12        LD    (DE),A
D7B3  13        INC   DE                Zeiger nach Offset auf Namen
D7B4  C9        RET

**********************************   eine Variable dimensionieren
D7B5  CD 06 D9  CALL  D906              Variablennamen u. Offset holen
D7B8  7E        LD    A,(HL)            folgendes Zeichen
D7B9  FE 28     CP    28                Klammer auf ?
D7BB  28 05     JR    Z,D7C2            dann o.k.
D7BD  EE 5B     XOR   5B                eckige Klammer auf ?
D7BF  C2 42 D6  JP    NZ,D642           sonst "Syntax error"
```

```
D7C2  CD 5A D8    CALL   D85A         Indizes holen, auf Basic-Stack
D7C5  E5          PUSH   HL           Basic-PC
D7C6  C5          PUSH   BC           und Zahl d. Dimensionen retten
D7C7  3A C1 B0    LD     A,(B0C1)     Typ der Variablen
D7CA  CD EA D5    CALL   D5EA         Adr. 1. Offs. d. VL, Basisadr.
D7CD  CD 08 D7    CALL   D708         Feldvariable suchen
D7D0  DA 4A D6    JP     C,D64A       gefunden ? dann Fehler
D7D3  C1          POP    BC           Zahl der Dimensionen zurück
D7D4  3E FF       LD     A,FF         Flag für kein Default-DIM
D7D6  CD 8A D8    CALL   D88A         Feldvariable dimensionieren
D7D9  E1          POP    HL           Basic-PC zurück
D7DA  C9          RET

***********************************  Var.-Adr. holen, auf Feld prüfen
                                     IN : CY=1 ,wenn  Offset eingetr.
                                          CY=1: DE: Feld/Var.-Offset
                                          CY=0: DE: Zeiger Var.-Token
OUT: CY=1, wenn Var. gefunden
                                          CY=1: DE: Feld/Var.-Adresse
                                          CY=0: DE wie IN
D7DB  F5          PUSH   AF           Flag für Offset eingetragen
D7DC  7E          LD     A,(HL)       Zeichen aus Basic-Text
D7DD  FE 28       CP     28           Klammer auf ?
D7DF  28 10       JR     Z,D7F1       dann Feldvariable
D7E1  EE 5B       XOR    5B           eckige Klammer auf ?
D7E3  28 0C       JR     Z,D7F1       dann Feldvariable
D7E5  F1          POP    AF           Flag für Offset zurück
D7E6  D0          RET    NC           Offset n. eingtr. ? dann zur.
D7E7  E5          PUSH   HL           Basic-PC retten
D7E8  2A 85 AE    LD     HL,(AE85)    Zeiger auf Variablenstart
D7EB  2B          DEC    HL           -1, Korrektur für Offset
D7EC  19          ADD    HL,DE        Offset addieren
D7ED  EB          EX     DE,HL        gibt Variablenadr., nach DE
D7EE  E1          POP    HL           Basic-PC zurück
D7EF  37          SCF                 CY=1 für gefunden
D7F0  C9          RET

D7F1  CD 5A D8    CALL   D85A         Indizes holen, auf Basic-Stack
D7F4  F1          POP    AF           Flag für Offset
D7F5  E5          PUSH   HL           Basic-PC retten
D7F6  30 07       JR     NC,D7FF      Offs. n. einget. ? d. Feld su.
D7F8  2A 87 AE    LD     HL,(AE87)    Zeiger auf Start der Felder
D7FB  2B          DEC    HL           -1, Korrektur für Offset
D7FC  19          ADD    HL,DE        Offset addieren
D7FD  18 15       JR     D814         Adresse des Feldelements ber.
D7FF  C5          PUSH   BC           Zahl der Dimensionen
D800  D5          PUSH   DE           und Zeiger auf Var.-Token
D801  3A C1 B0    LD     A,(B0C1)     Typ des Feldes
D804  CD EA D5    CALL   D5EA         Adr. d. 1. Offsets d. VL holen
D807  CD 08 D7    CALL   D708         Feld in VL suchen
D80A  30 0F       JR     NC,D81B      nicht gef. ? d. dimensionieren
D80C  13          INC    DE             Zeiger auf
D80D  13          INC    DE             Zahl der Dimensionen
D80E  E1          POP    HL           Zeiger auf Variablen-Token
D80F  CD 6D D7    CALL   D76D         Offset ins Programm eintragen
D812  C1          POP    BC           Zahl der Dimensionen
D813  EB          EX     DE,HL        Zg. auf Zahl d. Dimens. n. HL
D814  78          LD     A,B          Dimensionen-Zahl im Programm
```

```
D815  96          SUB   (HL)        = Zahl der Dimens. im Feld ?
D816  C2 46 D6    JP    NZ,D646     nein ? dann Fehler
D819  18 0A       JR    D825        Adresse des Feldelements ber.
D81B  E1          POP   HL          Zeiger auf Variablen-Token
D81C  C1          POP   BC          Zahl der Dimensionen
D81D  AF          XOR   A           Flag, Default-Dimensionierung
D81E  CD 8A D8    CALL  D88A        Feld dimensionieren
D821  CD 6D D7    CALL  D76D        Offset in Programm eintragen
D824  EB          EX    DE,HL       Zg. auf Dimens.-Zahl nach HL
D825  11 00 00    LD    DE,0000     Nr. des Feldelements=0
D828  46          LD    B,(HL)      Zahl der Dimensionen
D829  23          INC   HL          Zeiger auf Indextabelle
D82A  E5          PUSH  HL          retten
D82B  D5          PUSH  DE          Nr. des Feldelements
D82C  5E          LD    E,(HL)
D82D  23          INC   HL            nächsten Index aus Tabelle,
D82E  56          LD    D,(HL)        nach DE
D82F  3E 02       LD    A,02        2 Bytes (für nächsten Index)
D831  CD A0 F5    CALL  F5A0        vom Basic-Stack
D834  7E          LD    A,(HL)        aktuellen Index vom
D835  23          INC   HL            Basic-Stack holen,
D836  66          LD    H,(HL)        nach HL
D837  6F          LD    L,A
D838  CD B8 FF    CALL  FFB8        m. maximalem Index vergleichen
D83B  D2 46 D6    JP    NC,D646     akt. Index zu groß ? d. Fehler
D83E  E3          EX    (SP),HL     Index retten, Elementnr. n. HL
D83F  CD BE BD    CALL  BDBE        mit max. Index multiplizieren
D842  D1          POP   DE          aktuellen Index
D843  19          ADD   HL,DE       addieren
D844  EB          EX    DE,HL       neue Elementnr. nach DE
D845  E1          POP   HL          Zeiger in Indextabelle
D846  23          INC   HL            Zeiger auf nächsten
D847  23          INC   HL            Index
D848  05          DEC   B           Zahl der Dimensionen
D849  20 DF       JR    NZ,D82A     weitere Indizes ?
D84B  E5          PUSH  HL          Zeiger auf Feldelemente
D84C  2A C1 B0    LD    HL,(B0C1)   Typflag/Größe eines Elements
D84F  26 00       LD    H,00        Größe hi=0
D851  CD BE BD    CALL  BDBE        mit Nr. des Elements multipl.
D854  D1          POP   DE          Zeiger auf Feldelemente
D855  19          ADD   HL,DE       Offset für Element addieren
D856  EB          EX    DE,HL       Adresse des Elements nach DE
D857  E1          POP   HL          Basic-PC zurück
D858  37          SCF               CY=1 für gefunden
D859  C9          RET

*********************************  Indizes holen, auf Basic-Stack
                                 OUT: B: Zahl der Indizes
D85A  D5          PUSH  DE
D85B  CD 3F DD    CALL  DD3F          nächstes Zeichen
D85E  3A C1 B0    LD    A,(B0C1)    Typ der Variablen
D861  F5          PUSH  AF          retten
D862  06 00       LD    B,00        Zahl der Indizes=0
D864  CD 7C CE    CALL  CE7C        nächsten Index holen
D867  E5          PUSH  HL          Basic-PC retten
D868  3E 02       LD    A,02        2 Bytes für Index
D86A  CD B0 F5    CALL  F5B0        auf Basic-Stack reservieren
D86D  73          LD    (HL),E
```

```
D86E  23          INC   HL           Index auf Basic-Stack
D86F  72          LD    (HL),D
D870  E1          POP   HL          Basic-PC zurück
D871  04          INC   B           Zahl der Indizes erhöhen
D872  CD 55 DD    CALL  DD55        folgt Komma ?
D875  38 ED       JR    C,D864      dann nächster Index
D877  7E          LD    A,(HL)      nächstes Zeichen
D878  FE 29       CP    29          Klammer zu ?
D87A  28 05       JR    Z,D881      dann o.k.
D87C  FE 5D       CP    5D          eckige Klammer zu ?
D87E  C2 42 D6    JP    NZ,D642     nein ? dann "Syntax error"
D881  CD 3F DD    CALL  DD3F        Klammer übergehen
D884  F1          POP   AF          Typ der Feldvariablen
D885  32 C1 B0    LD    (B0C1),A    wieder setzen
D888  D1          POP   DE
D889  C9          RET

**********************************  Feldvariable neu anlegen
                                    IN : A=0 für Default-DIM
                                         A=$FF für Index-DIM
                                         B: Zahl der Dimensionen
                                    OUT: DE: Zeiger auf Zahl der Di-
                                             mensionen im Feldeintrag
                                         BC: Start der Felder-1
D88A  E5          PUSH  HL
D88B  32 26 AE    LD    (AE26),A    Flag für Default-DIM retten
D88E  C5          PUSH  BC          Zahl der Dimensionen
D88F  78          LD    A,B         nach A
D890  87          ADD   A           2 Bytes pro Index
D891  C6 03       ADD   03          Platz f. Dim.-Zahl/Feldlänge
D893  CD 77 D7    CALL  D777        Namenlänge und ben. Platz hol.
D896  F5          PUSH  AF          Länge des Namens retten
D897  2A 89 AE    LD    HL,(AE89)   Zeiger auf Ende der Felder
D89A  EB          EX    DE,HL       als Einfügestelle nach DE
D89B  CD F8 F5    CALL  F5F8        Platz reservieren
D89E  F1          POP   AF          Länge des Namens
D89F  CD 8A D7    CALL  D78A        Namen und Typ übertragen
D8A2  60          LD    H,B           Zeiger nach Typflag
D8A3  69          LD    L,C           nach HL
D8A4  C1          POP   BC          Zahl der Dimensionen
D8A5  D5          PUSH  DE          Zeiger auf Feldeintrag
D8A6  23          INC   HL            Zeiger auf Typ+2, Platz
D8A7  23          INC   HL            für Länge übergehen
D8A8  3A C1 B0    LD    A,(B0C1)    Typ des Feldes
D8AB  5F          LD    E,A           als Feldgröße
D8AC  16 00       LD    D,00          nach DE
D8AE  70          LD    (HL),B      Dim.-Zahl in Feld speichern
D8AF  E5          PUSH  HL          Zeiger auf Zahl der Dimens.
D8B0  23          INC   HL          Zeiger auf Indextabelle
D8B1  D5          PUSH  DE          Feldgröße retten
D8B2  3A 26 AE    LD    A,(AE26)    Flag für Default-DIM
D8B5  B7          OR    A           Default-Dimensionierung ?
D8B6  11 0B 00    LD    DE,000B     11 als Default-Index
D8B9  28 0B       JR    Z,D8C6      ggf. Default als Index
D8BB  E5          PUSH  HL          Zeiger in Indextabelle retten
D8BC  3E 02       LD    A,02        2 Bytes für nächsten Index
D8BE  CD A0 F5    CALL  F5A0        vom Basic-Stack
D8C1  5E          LD    E,(HL)
```

```
D8C2  23          INC   HL            Index aus Basic-Stack
D8C3  56          LD    D,(HL)        laden, nach DE
D8C4  13          INC   DE
D8C5  E1          POP   HL           Zeiger in Indextabelle
D8C6  73          LD    (HL),E
D8C7  23          INC   HL            Index in Indextabelle
D8C8  72          LD    (HL),D        des Feldes eintragen
D8C9  23          INC   HL
D8CA  E3          EX    (SP),HL      Tab.-Zg. retten, Feldgr. n. HL
D8CB  CD BE BD    CALL  BDBE         bisherige Feldgröße mal Index
D8CE  DA 46 D6    JP    C,D646       Übertrag ? dann Fehler
D8D1  EB          EX    DE,HL        neue Feldgröße nach DE
D8D2  E1          POP   HL           Zeiger in Indextabelle
D8D3  10 DC       DJNZ  D8B1         weitere Indizes ?
D8D5  42          LD    B,D           Feldgröße
D8D6  4B          LD    C,E           als benötigter Platz nach BC
D8D7  54          LD    D,H           Zeiger nach Indextabelle
D8D8  5D          LD    E,L           als Einfügeadresse nach DE
D8D9  CD FB F5    CALL  F5FB         Platz reservieren
D8DC  22 89 AE    LD    (AE89),HL    neues Ende der Felder setzen
D8DF  C5          PUSH  BC           Länge der Feldelemente retten
D8E0  2B          DEC   HL
D8E1  36 00       LD    (HL),00      Feld-Byte löschen
D8E3  0B          DEC   BC           Bytezähler
D8E4  78          LD    A,B
D8E5  B1          OR    C
D8E6  20 F8       JR    NZ,D8E0      weitere Bytes ?
D8E8  C1          POP   BC           Länge der Feldelemente
D8E9  E1          POP   HL           Zeiger auf Zahl der Dimens.
D8EA  5E          LD    E,(HL)       Dimensionen-Zahl lo
D8EB  16 00       LD    D,00         hi=0
D8ED  EB          EX    DE,HL        nach HL, Zg. auf Dim. nach DE
D8EE  29          ADD   HL,HL        2 Bytes pro Index
D8EF  23          INC   HL           +1 für Dimensionsbyte
D8F0  09          ADD   HL,BC        Länge d. Feldelemente addieren
D8F1  EB          EX    DE,HL        Feldlänge nach DE
D8F2  2B          DEC   HL            Zeiger auf Dimensionsbyte -2
D8F3  2B          DEC   HL            gibt Zeiger a. Längeneintrag
D8F4  73          LD    (HL),E
D8F5  23          INC   HL            Feldlänge eintragen
D8F6  72          LD    (HL),D
D8F7  23          INC   HL           Zeiger auf Dimensionsbyte
D8F8  E3          EX    (SP),HL      retten, Zg. a. Feldeintr. zur.
D8F9  EB          EX    DE,HL        nach DE, Feldlänge nach HL
D8FA  3A C1 B0    LD    A,(B0C1)     Typ des Feldes
D8FD  CD EA D5    CALL  D5EA         Adr. d. 1. Offsets d. VL holen
D900  CD A5 D7    CALL  D7A5         Feld in entspr. VL einhängen
D903  D1          POP   DE           Zeiger auf Dimensionsbyte
D904  E1          POP   HL
D905  C9          RET

*********************************   Variablenname und Offset holen
                                    OUT: CY=1: Offset eingetragen
                                         DE: Offset für Variable
                                         CY=0: Offset nicht eingetr.
                                         DE: Zeiger auf Var.-Token
                                         C: 1. Byte des Namens
                                         ($AE27): Zeiger auf Namen
```

```
D906  CD 7F D9    CALL   D97F         Typ nach Variablen-Token setz.
D909  23          INC    HL           Zeiger nach Token
D90A  5E          LD     E,(HL)
D90B  23          INC    HL             Offset der Variable holen
D90C  56          LD     D,(HL)
D90D  7A          LD     A,D
D90E  B3          OR     E            Offset nicht eingetragen ?
D90F  28 0A       JR     Z,D91B       dann Namen auswerten
D911  23          INC    HL
D912  7E          LD     A,(HL)       Zeichen aus Name
D913  17          RLA                 Ende des Namen ?
D914  30 FB       JR     NC,D911      nein ? dann weiter überlesen
D916  CD 3F DD    CALL   DD3F         Zeichen nach Name
D919  37          SCF                 CY=1 für Offset eingetragen
D91A  C9          RET

D91B  2B          DEC    HL
D91C  2B          DEC    HL             PC zurück auf Token
D91D  EB          EX     DE,HL        nach DE
D91E  C1          POP    BC           Aufrufadresse
D91F  2A 27 AE    LD     HL,(AE27)    alten Variablennamen-Zeiger
D922  E5          PUSH   HL           retten
D923  21 2B D9    LD     HL,D92B      Namen von Basic-Stack löschen
D926  E5          PUSH   HL           als Rücksprungadresse
D927  C5          PUSH   BC           alte Rücksprungadresse
D928  EB          EX     DE,HL        Zeiger auf Var.-Token nach HL
D929  18 0E       JR     D939         Namen holen, auf Basic-Stack

*********************************** Variablennamen vom Basic-Stack
D92B  E5          PUSH   HL
D92C  2A 27 AE    LD     HL,(AE27)    Zeiger auf Namen
D92F  CD AC F5    CALL   F5AC         als Basic-SP setz. (Namen lö.)
D932  E1          POP    HL             zweitobersten
D933  E3          EX     (SP),HL        Stackeintrag holen
D934  22 27 AE    LD     (AE27),HL    alten Namenzeiger zurück
D937  E1          POP    HL
D938  C9          RET

*********************************** Variablennamen auf Basic-Stack
D939  E5          PUSH   HL           PC auf Variablen-Token
D93A  7E          LD     A,(HL)       Token
D93B  23          INC    HL
D93C  23          INC    HL             Offset übergehen
D93D  23          INC    HL
D93E  4E          LD     C,(HL)       1. Zeichen aus Namen
D93F  CB A9       RES    5,C          auf Großschrift forcieren
D941  FE 0B       CP     0B           markierte Variable ?
D943  38 19       JR     C,D95E       dann keine Typenbestimmung
D945  79          LD     A,C          1. Zeichen des Namens
D946  E6 1F       AND    1F           Nr. des Buchstaben
D948  C6 0B       ADD    0B
D94A  5F          LD     E,A            $AE0B ($AE0C-1, Adresse der
D94B  CE AE       ADC    AE             Tabelle für DEFINT/STR/REAL)
D94D  93          SUB    E              addieren
D94E  57          LD     D,A
D94F  1A          LD     A,(DE)       Typflag aus Tabelle laden
D950  32 C1 B0    LD     (B0C1),A     als Typ der Variablen setzen
D953  E3          EX     (SP),HL      Adr. Namen r., Adr. Token zur.
```

```
D954  36 0D     LD    (HL),0D      Token f. REAL-Var. ohne Kennz.
D956  FE 05     CP    05           REAL-Variable ?
D958  28 03     JR    Z,D95D       dann Token o.k.
D95A  C6 09     ADD   09             sonst Tok. f. String/Integer
D95C  77        LD    (HL),A         ohne Kennz. setzen
D95D  E3        EX    (SP),HL      Adr. Token r., Adr. Namen zur.
D95E  EB        EX    DE,HL        Zeiger auf Name nach DE
D95F  3E 28     LD    A,28         40 Bytes für Namen
D961  CD B0 F5  CALL  F5B0         Platz auf Basic-Stack reserv.
D964  22 27 AE  LD    (AE27),HL    Zeiger auf Platz speichern
D967  06 29     LD    B,29         max. Namenlänge+1
D969  05        DEC   B            restl. Namenbytes
D96A  CA 42 D6  JP    Z,D642       zu lang ? dann "Syntax error"
D96D  1A        LD    A,(DE)       Byte aus Namen
D96E  13        INC   DE
D96F  E6 DF     AND   DF           auf Großschrift forcieren
D971  77        LD    (HL),A       in Basic-Stack übertragen
D972  23        INC   HL
D973  17        RLA                Ende des Namens erreicht ?
D974  30 F3     JR    NC,D969      nein ? dann weiter
D976  CD AC F5  CALL  F5AC         Namenende als neuen Basic-SP
D979  EB        EX    DE,HL        Zeiger nach Namen nach HL
D97A  2B        DEC   HL           Zeiger auf letztes Namenbyte
D97B  D1        POP   DE           Zeiger auf Var.-Token zurück
D97C  C3 3F DD  JP    DD3F         Zeichen nach Namen holen

********************************** Variablentyp entspr. Token setzen
D97F  7E        LD    A,(HL)         Variablen-Token
D980  FE 0B     CP    0B             Var. m. Kennz./Statementende ?
D982  38 02     JR    C,D986         dann auswerten
D984  C6 F7     ADD   F7             Token f. Var. m. K. generieren
D986  FE 04     CP    04             REAL-Variable ?
D988  28 09     JR    Z,D993         dann Typ auf REAL setzen
D98A  30 04     JR    NC,D990        keine Variable ? dann Fehler
D98C  FE 02     CP    02             Statementende ?
D98E  30 05     JR    NC,D995        nein ? dann Typ setzen
D990  C3 42 D6  JP    D642           "Syntax error"
D993  3E 05     LD    A,05           Typ für REAL
D995  32 C1 B0  LD    (B0C1),A       Typ setzen
D998  C9        RET

********************************** VL der Felder neu generieren
D999  CD C6 D5  CALL  D5C6           VL der Felder löschen
D99C  2A 89 AE  LD    HL,(AE89)      Zeiger auf Ende der Felder
D99F  EB        EX    DE,HL          nach DE
D9A0  2A 87 AE  LD    HL,(AE87)      Zeiger auf Start der Felder
D9A3  CD B8 FF  CALL  FFB8           akt. Zeiger = Ende d. Felder ?
D9A6  C8        RET   Z              dann fertig
D9A7  D5        PUSH  DE             Zeiger auf Ende der Felder
D9A8  CD 31 D7  CALL  D731           Feldnamen überlesen
D9AB  7E        LD    A,(HL)         Typflag des Feldes
D9AC  23        INC   HL             Zeiger nach Typflag
D9AD  E6 07     AND   07               Typflag nach FAC-
D9AF  3C        INC   A                Typflag-Format
D9B0  E5        PUSH  HL             Zeiger nach Typflag
D9B1  CD EA D5  CALL  D5EA           Adr. d. 1. Offset der VL holen
D9B4  CD A5 D7  CALL  D7A5           Feld in entspr. VL einhängen
D9B7  E1        POP   HL             Zeiger nach Typflag
```

```
D9B8  5E          LD     E,(HL)
D9B9  23          INC    HL            Feldlänge laden,
D9BA  56          LD     D,(HL)        nach DE
D9BB  23          INC    HL
D9BC  19          ADD    HL,DE       addieren, gibt Adr. d. n. Feld
D9BD  D1          POP    DE          Zeiger auf Ende der Felder
D9BE  18 E3       JR     D9A3        nächstes Feld einhängen

*********************************** Basic-Befehl ERASE
D9C0  CD 89 E9    CALL   E989        Var.-Offsets im Prg. löschen
D9C3  CD CC D9    CALL   D9CC        ein Feld löschen
D9C6  CD 55 DD    CALL   DD55        folgt Komma ?
D9C9  38 F8       JR     C,D9C3      dann nächstes Feld
D9CB  C9          RET

*********************************** ein Feld löschen
D9CC  CD 06 D9    CALL   D906        Variablennamen u. Offset holen
D9CF  E5          PUSH   HL          Basic-PC retten
D9D0  3A C1 B0    LD     A,(B0C1)    Typ der Variablen
D9D3  CD EA D5    CALL   D5EA        Adr. d. Offsets der VL holen
D9D6  CD 08 D7    CALL   D708        Feld in entspr. VL suchen
D9D9  E5          PUSH   HL          Zeiger auf Feldeintrag
D9DA  EB          EX     DE,HL       Zeiger nach Typ nach HL
D9DB  1E 05       LD     E,05        Nr. für "Improper argument"
D9DD  D2 94 CA    JP     NC,CA94     Feld nicht gef. ? dann Fehler
D9E0  5E          LD     E,(HL)
D9E1  23          INC    HL            Länge des Feldes laden,
D9E2  56          LD     D,(HL)        nach DE
D9E3  23          INC    HL
D9E4  19          ADD    HL,DE       addieren
D9E5  EB          EX     DE,HL       gibt Feld-Endadresse, nach DE
D9E6  2A 89 AE    LD     HL,(AE89)   Zeiger auf Ende der Felder
D9E9  CD CF FF    CALL   FFCF        Feld-Endadresse abziehen
D9EC  E3          EX     (SP),HL     Länge bis Ende r., Feldadr. z.
D9ED  C1          POP    BC          zu verschiebende Länge
D9EE  EB          EX     DE,HL       Feldadr. n. DE, Endadr. n. HL
D9EF  78          LD     A,B           Länge <>0 ?
D9F0  B1          OR     C
D9F1  C4 F2 FF    CALL   NZ,FFF2     dann folgende Felder versch.
D9F4  EB          EX     DE,HL       Endadresse des versch. Blocks
D9F5  22 89 AE    LD     (AE89),HL   als neues Ende der Felder
D9F8  CD 99 D9    CALL   D999        VL der Felder neu generieren
D9FB  E1          POP    HL          Basic-PC zurück
D9FC  C9          RET

*********************************** FN-Listenzeiger löschen
D9FD  21 00 00    LD     HL,0000
DA00  22 2B AE    LD     (AE2B),HL   Zeiger auf FN-VL
DA03  22 29 AE    LD     (AE29),HL   Zeiger a. akt. FN-Listeneintr.
DA06  C9          RET

*********************************** neuen Eintrag in FN-Liste gener.
DA07  E5          PUSH   HL
DA08  2A 2B AE    LD     HL,(AE2B)   Zeiger auf 1. Eintrag d. FN-VL
DA0B  E5          PUSH   HL          retten
DA0C  2A 29 AE    LD     HL,(AE29)   Zeiger a. bearbeiteten Eintrag
DA0F  EB          EX     DE,HL       nach DE
DA10  3E 06       LD     A,06        Größe eines FN-Listeneintrags
```

```
DA12  CD B0 F5    CALL   F5B0          Platz auf Basic-Stack reserv.
DA15  22 29 AE    LD     (AE29),HL     als Zeiger auf akt. Eintrag
DA18  73          LD     (HL),E
DA19  23          INC    HL              Zeiger auf vorher bearbei-
DA1A  72          LD     (HL),D          teten Eintrag eintragen
DA1B  23          INC    HL
DA1C  AF          XOR    A
DA1D  77          LD     (HL),A          Zeiger auf VL der Funktions-
DA1E  23          INC    HL              variablen dieses FN-Listen-
DA1F  77          LD     (HL),A          eintrags löschen
DA20  23          INC    HL
DA21  D1          POP    DE
DA22  73          LD     (HL),E
DA23  23          INC    HL              Kettungsadresse für
DA24  72          LD     (HL),D          FN-Liste eintragen
DA25  E1          POP    HL
DA26  C9          RET

*********************************** Eintrag in FN-Liste einhängen
DA27  E5          PUSH   HL
DA28  2A 29 AE    LD     HL,(AE29)     akt. berarbeiteten Eintrag
DA2B  22 2B AE    LD     (AE2B),HL     als 1. Eintrag der FN-Liste
DA2E  E1          POP    HL
DA2F  C9          RET

*********************************** Eintrag aus FN-Liste aushängen
DA30  E5          PUSH   HL
DA31  2A 29 AE    LD     HL,(AE29)     Zeiger auf akt. Eintrag
DA34  CD AC F5    CALL   F5AC          Eintrag v. Basic-Stack löschen
DA37  5E          LD     E,(HL)
DA38  23          INC    HL              vorher bearbeiteten
DA39  56          LD     D,(HL)          Listeneintrag
DA3A  23          INC    HL
DA3B  EB          EX     DE,HL
DA3C  22 29 AE    LD     (AE29),HL       als akt. Eintrag setzen
DA3F  EB          EX     DE,HL
DA40  23          INC    HL              Zeiger auf Funktionsvaria-
DA41  23          INC    HL              blenliste übergehen
DA42  5E          LD     E,(HL)
DA43  23          INC    HL              nächsten Eintrag d. FN-Liste
DA44  56          LD     D,(HL)          (Kettungsadresse)
DA45  EB          EX     DE,HL
DA46  22 2B AE    LD     (AE2B),HL     als 1. Listeneintrag setzen
DA49  E1          POP    HL
DA4A  C9          RET

*********************************** Funktionsvar. holen, in VL eintr.
                                   OUT: DE: Adr. der Funktionsvar.
                                          B: Typflag
DA4B  E5          PUSH   HL            Funktionsdef.-PC retten
DA4C  3E 02       LD     A,02          2 Bytes für Kettungsadresse
DA4E  CD B0 F5    CALL   F5B0          auf Basic-Stack reservieren
DA51  E3          EX     (SP),HL       Zg. Stackeintr. retten, PC z.
DA52  CD 7F D9    CALL   D97F          Typ entspr. FN-Var.-Token
DA55  CD 39 D9    CALL   D939          FN-Var.-Namen auf Basic-Stack
DA58  E3          EX     (SP),HL       PC retten, Zeiger auf Stack-
DA59  EB          EX     DE,HL         eintrag nach DE
DA5A  2A 29 AE    LD     HL,(AE29)     Zg. auf akt. FN-Listeneintrag
```

```
DA5D  23          INC   HL            +2=Zeiger auf 1. Zeiger der
DA5E  23          INC   HL            Funktionsvariablen-VL
DA5F  01 00 00    LD    BC,0000      Basisadr.=0, da absolute Adr.
DA62  CD A5 D7    CALL  D7A5         FN-Var. in VL einhängen
DA65  3A C1 B0    LD    A,(B0C1)     Typ der Funktionsvariablen
DA68  47          LD    B,A
DA69  3C          INC   A            +1=benötigter Platz
DA6A  CD B0 F5    CALL  F5B0         Platz auf Basic-Stack reserv.
DA6D  78          LD    A,B          Typflag
DA6E  3D          DEC   A            nach Variablenformat
DA6F  77          LD    (HL),A       in Stackeintrag speichern
DA70  23          INC   HL           Zeiger auf FN-Variablenwert
DA71  EB          EX    DE,HL        nach DE
DA72  E1          POP   HL           Funkionsdef.-PC zurück
DA73  C9          RET

***********************************  sämtl. Stringvariablen durchgehen
                                     IN : DE: Bearbeitungsroutinenadr.
DA74  2A 29 AE    LD    HL,(AE29)    Zeiger a. akt. FN-Listeneintr.
DA77  7C          LD    A,H
DA78  B5          OR    L
DA79  28 0E       JR    Z,DA89       FN-Liste zu Ende ?
DA7B  4E          LD    C,(HL)
DA7C  23          INC   HL             sonst Adresse des nächsten
DA7D  46          LD    B,(HL)         Listeneintrags
DA7E  23          INC   HL
DA7F  C5          PUSH  BC             retten
DA80  01 00 00    LD    BC,0000      Basisadr.=0, da absolute Adr.
DA83  CD CE DA    CALL  DACE         FN-Var.-VL dies. Eint. durchg.
DA86  E1          POP   HL           Adr. des nächsten Eintrags
DA87  18 EE       JR    DA77         nächsten Listeneintrag bearb.
DA89  01 41 1A    LD    BC,1A41      26 Buchstaben, 1. Buchst.="A"
DA8C  C5          PUSH  BC           Zähler/Buchstaben retten
DA8D  79          LD    A,C          akt. Buchstabe
DA8E  CD DB D5    CALL  D5DB         1. Offset der entspr. VL holen
DA91  CD CE DA    CALL  DACE         diese VL durchgehen
DA94  C1          POP   BC           Zähler/akt. Buchstabe
DA95  0C          INC   C            nächster Buchstabe
DA96  05          DEC   B            Zähler
DA97  20 F3       JR    NZ,DA8C      weitere Buchstaben ?
DA99  3E 03       LD    A,03         Typflag für String
DA9B  CD EA D5    CALL  D5EA         1. Offset d. VL d. Str.-Felder
DA9E  4E          LD    C,(HL)
DA9F  23          INC   HL             Offset für nächstes String-
DAA0  46          LD    B,(HL)         feld laden
DAA1  78          LD    A,B
DAA2  B1          OR    C
DAA3  C8          RET   Z            VL zu Ende ? dann fertig
DAA4  2A 87 AE    LD    HL,(AE87)    Zeiger auf Start der Felder
DAA7  2B          DEC   HL           -1, Korrektur für Offset
DAA8  09          ADD   HL,BC        zu Basisadresse addieren
DAA9  E5          PUSH  HL           Adresse des Feldes retten
DAAA  D5          PUSH  DE
DAAB  CD 31 D7    CALL  D731         Feldnamen übergehen
DAAE  D1          POP   DE
DAAF  23          INC   HL           Typflag übergehen
DAB0  4E          LD    C,(HL)
DAB1  23          INC   HL           Länge der Feldelemente laden
```

```
DAB2  46          LD    B,(HL)
DAB3  23          INC   HL
DAB4  E5          PUSH  HL          Zeiger auf Dimensionsbyte
DAB5  09          ADD   HL,BC       Länge addieren, gibt Feldende
DAB6  E3          EX    (SP),HL     retten, Zg. Dimensionsbyte zu.
DAB7  4E          LD    C,(HL)      Zahl der Dimensionen
DAB8  23          INC   HL          Zeiger auf Indextabelle
DAB9  06 00       LD    B,00        Zahl der Dimensionen hi=0
DABB  09          ADD   HL,BC         2 mal addieren, da
DABC  09          ADD   HL,BC         2 Bytes pro Index
DABD  C1          POP   BC          Zeiger auf Ende des Feldes
DABE  CD BE FF    CALL  FFBE        Ende erreicht ?
DAC1  28 08       JR    Z,DACB      dann nächstes Feld
DAC3  CD E7 DA    CALL  DAE7        Bearbeitungsroutine ausf.
DAC6  23          INC   HL            Länge eines Descriptors
DAC7  23          INC   HL            addieren, gibt Zeiger auf
DAC8  23          INC   HL            nächstes Feldelement
DAC9  18 F3       JR    DABE        nächstes Element behandeln
DACB  E1          POP   HL          Adresse des Feldes
DACC  18 D0       JR    DA9E        nächstes Feld behandeln

*********************************  VL durchgehen, Routine ausführen
                                   IN : BC: Basisadresse; DE: Routinenadresse
                                        HL: Zeiger auf 1. Offset

DACE  7E          LD    A,(HL)
DACF  23          INC   HL            nächsten Offset laden
DAD0  66          LD    H,(HL)
DAD1  6F          LD    L,A
DAD2  B4          OR    H           VL zu Ende ?
DAD3  C8          RET   Z           dann fertig
DAD4  09          ADD   HL,BC       Basisadresse addieren
DAD5  E5          PUSH  HL          Zeiger auf Variable
DAD6  D5          PUSH  DE
DAD7  CD 31 D7    CALL  D731        Namen überlesen
DADA  D1          POP   DE
DADB  7E          LD    A,(HL)      Typflag der Variablen
DADC  23          INC   HL          Zeiger auf Variablenwert
DADD  E6 07       AND   07          FN-Kennz. löschen
DADF  FE 02       CP    02          Stringvariable ?
DAE1  CC E7 DA    CALL  Z,DAE7      dann Routine ausführen
DAE4  E1          POP   HL          Adresse des Variableneintrags
DAE5  18 E7       JR    DACE        nächste Variable in VL

*********************************  Stringbearbeitungsroutine ausf.
                                   IN : HL: Zeiger auf Descriptor
                                        DE: Routinenadresse

DAE7  C5          PUSH  BC
DAE8  D5          PUSH  DE
DAE9  E5          PUSH  HL
DAEA  7E          LD    A,(HL)      Länge
DAEB  23          INC   HL
DAEC  4E          LD    C,(HL)        und Stringadresse
DAED  23          INC   HL            aus Descriptor laden
DAEE  46          LD    B,(HL)
DAEF  EB          EX    DE,HL       Rout.-Adr. HL, Descr.-Ende DE
DAF0  B7          OR    A           Länge <>0 ?
DAF1  C4 F8 FF    CALL  NZ,FFF8     dann Routine ausführen
DAF4  E1          POP   HL
```

```
DAF5  D1          POP    DE
DAF6  C1          POP    BC
DAF7  C9          RET

*********************************  Basic-Befehl LINE INPUT
DAF8  CD 37 DD    CALL   DD37        Test auf INPUT
DAFB  A3                             Token für INPUT
DAFC  CD CB C1    CALL   C1CB        opt. Filenr. als Eingabekanal
DAFF  F5          PUSH   AF          alte Kanalnr. retten
DB00  CD 89 DB    CALL   DB89        ggf. Text ausgeben
DB03  CD 86 D6    CALL   D686        Variable holen
DB06  CD 3C FF    CALL   FF3C        Test auf Stringvariable
DB09  E5          PUSH   HL          Basic-PC
DB0A  D5          PUSH   DE          und Zeiger auf Variable retten
DB0B  CD 1A DB    CALL   DB1A        Eingabezeile holen
DB0E  CD DC F7    CALL   F7DC        als String auf Stringstack
DB11  E1          POP    HL          Adresse der Variablen
DB12  CD 6F D6    CALL   D66F        String an Variable zuweisen
DB15  E1          POP    HL          Basic-PC
DB16  F1          POP    AF          alte Kanalnr.
DB17  C3 AF C1    JP     C1AF        wieder als Eingabekanal

*********************************  Zeile für LINE INPUT holen
                                   OUT: HL: Zeiger auf Zeile
DB1A  CD C0 C1    CALL   C1C0        akt. Eingabekanalnr. holen
DB1D  D2 66 DC    JP     NC,DC66     Kassette ?
DB20  CD A2 C1    CALL   C1A2        sonst Nr. als Streamnr. setzen
DB23  F5          PUSH   AF          alte Streamnr. retten
DB24  CD AD DB    CALL   DBAD        Eingabezeile v. Tastatur holen
DB27  F1          POP    AF          alte Streamnr.
DB28  C3 A2 C1    JP     C1A2        wieder setzen

*********************************  Basic-Befehl INPUT
DB2B  CD CB C1    CALL   C1CB        opt. Filenr. als Eingabekanal
DB2E  F5          PUSH   AF          alte Kanalnr. retten
DB2F  CD 47 DB    CALL   DB47        Eingabezeile holen und prüfen
DB32  D5          PUSH   DE          Zeiger auf Zeile retten
DB33  CD 86 D6    CALL   D686        Variable holen
DB36  E3          EX     (SP),HL     PC retten, Eingabezeiger zur.
DB37  3E 00       LD     A,00        Zeilenende als Trennzeichen
DB39  CD BC DB    CALL   DBBC        Eingabe an Variable zuweisen
DB3C  E3          EX     (SP),HL     Eingabezeiger retten, PC zur.
DB3D  CD 55 DD    CALL   DD55        folgt Komma ?
DB40  38 F1       JR     C,DB33      dann nächste Variable
DB42  D1          POP    DE          Eingabezeiger
DB43  F1          POP    AF          alte Kanalnr.
DB44  C3 AF C1    JP     C1AF        wieder als Eingabekanal setzen

*********************************  Eingabezeile holen und prüfen
                                   OUT: DE: Zeiger auf Zeile
DB47  CD C0 C1    CALL   C1C0        Eingabekanalnr. holen
DB4A  30 3D       JR     NC,DB89     Kassette ? dann nur Text ausg.
DB4C  CD A2 C1    CALL   C1A2        Nr. als Streamnr. setzen
DB4F  F5          PUSH   AF          alte Streamnr. retten
DB50  E5          PUSH   HL          PC für "Redo" retten
DB51  CD 89 DB    CALL   DB89        ggf. Text ausgeben, Flags hol.
DB54  3E 3F       LD     A,3F        "?"
DB56  D4 56 C3    CALL   NC,C356     Flag für "?" ? dann ausgeben
```

```
DB59  3E 20       LD     A,20         Space
DB5B  D4 56 C3    CALL   NC,C356      Flag für "?" ? dann ausgeben
DB5E  E5          PUSH   HL           PC retten
DB5F  CD AD DB    CALL   DBAD         Eingabezeile von Tastatur hol.
DB62  EB          EX     DE,HL        Zeiger auf Zeile nach DE
DB63  E1          POP    HL           Basic-PC
DB64  CD D3 DB    CALL   DBD3         Eingabezeile überprüfen
DB67  38 09       JR     C,DB72       kein Fehler ?
DB69  21 77 DB    LD     HL,DB77      Zeiger auf "?Redo from start"
DB6C  CD 41 C3    CALL   C341         String ausgeben
DB6F  E1          POP    HL           PC nach INPUT-Token
DB70  18 DE       JR     DB50         neue Zeile holen
DB72  F1          POP    AF           Zeiger für "Redo" löschen
DB73  F1          POP    AF           alte Streamnr.
DB74  C3 A2 C1    JP     C1A2         wieder setzen

************************************

DB77  3F 52 65 64 6F 20 66 72         ?Redo fr
DB7F  6F 6D 20 73 74 61 72 74         om start
DB87  0A 00                           ..

************************************ ggf. Text ausgeben, Flags holen
                                    OUT: CY=0 für "?" ausgeben
                                         ($AE2D): Flag für Linefeed
DB89  7E          LD     A,(HL)       Zeichen aus Basic-Text
DB8A  FE 3B       CP     3B           ";" ?
DB8C  32 2D AE    LD     (AE2D),A     als Flag für Linefeed setzen
DB8F  CC 3F DD    CALL   Z,DD3F       ";" ? dann ";" übergehen
DB92  EE 22       XOR    22           '"' ? (CY=0 für "?" ausgeben)
DB94  C0          RET    NZ           nein ? dann kein Text
DB95  CD CB F7    CALL   F7CB         String holen, auf Stringstack
DB98  CD C0 C1    CALL   C1C0         Eingabekanalnr. holen
DB9B  F5          PUSH   AF           und retten
DB9C  DC 28 F8    CALL   C,F828       nicht Kassette ? dann ausgeben
DB9F  F1          POP    AF
DBA0  D4 DA FB    CALL   NC,FBDA      Kassette ? dann Str. vom Stack
DBA3  CD 55 DD    CALL   DD55         folgt Komma ?
DBA6  D8          RET    C            dann zurück
DBA7  CD 37 DD    CALL   DD37         sonst Test auf ";"
DBAA  3B                              ";"
DBAB  B7          OR     A            CY=0 für "?" ausgeben
DBAC  C9          RET

************************************ Eingabezeile von Tastatur holen
                                    OUT: HL: Zeiger auf Zeile
DBAD  CD 3B CA    CALL   CA3B         Eingabezeile holen
DBB0  D2 6B CB    JP     NC,CB6B      Abbruch ? dann "Break"
DBB3  3A 2D AE    LD     A,(AE2D)     Flag für Linefeed
DBB6  FE 3B       CP     3B           gesetzt ?
DBB8  C4 4E C3    CALL   NZ,C34E      dann Linefeed ausgeben
DBBB  C9          RET

************************************ Eingabe an Variable zuweisen
                                    IN : DE: Variablenadresse
                                         HL: Eingabezeiger; A: Trennzeichen
                                    OUT: HL: Eingabezeiger
                                         nur beim CPC 464: Z=1 bei Komma/Zeilenende
DBBC  D5          PUSH   DE           Variablenadresse retten
```

```
DBBD  CD 02 DC     CALL   DC02        Eingabe auswerten/wandeln
DBC0  30 0C        JR     NC,DBCE     Fehler ? dann "Type mismatch"
DBC2  E3           EX     (SP),HL     Eingabezg. retten, Var.-Adr.
DBC3  CD 66 D6     CALL   D666        Wert an Variable zuweisen
DBC6  E1           POP    HL          Eingabezeiger
DBC7  7E           LD     A,(HL)      Zeichen nach Eingabewert
DBC8  23           INC    HL          Zeiger nach zugewiesener Eing.
DBC9  B7           OR     A
DBCA  C8           RET    Z           Zeilenende ?
DBCB  EE 2C        XOR    2C          Komma ?
DBCD  C9           RET

DBCE  1E 0D        LD     E,0D        Nr. für "Type mismatch"
DBD0  C3 94 CA     JP     CA94        Fehler ausgeben

*********************************** Eingabezeile überprüfen
                                    IN/OUT: DE: Zeiger auf Zeile
                                                HL: Basic-PC
                                    OUT: CY=0 bei Fehler
DBD3  D5           PUSH   DE
DBD4  E5           PUSH   HL
DBD5  D5           PUSH   DE          Zeiger auf Eingabe
DBD6  CD D6 D6     CALL   D6D6        Variable überlesen, Typ holen
DBD9  E3           EX     (SP),HL     PC retten, Eing.-Zg. zurück
DBDA  AF           XOR    A           Zeilenende als Trennzeichen
DBDB  CD 02 DC     CALL   DC02        Eingabe auswerten
DBDE  30 1E        JR     NC,DBFE     Fehler ? dann zurück
DBE0  FE 03        CP     03          Stringeingabe ?
DBE2  CC DA FB     CALL   Z,FBDA      dann vom Stringstack löschen
DBE5  E3           EX     (SP),HL
DBE6  CD 55 DD     CALL   DD55          folgt Komma im Programm ?
DBE9  E3           EX     (SP),HL
DBEA  30 0B        JR     NC,DBF7     nein ? dann fertig
DBEC  CD 61 DD     CALL   DD61        Spaces, TABs, LFs überlesen
DBEF  EE 2C        XOR    2C          Komma an akt. Eingabepos. ?
DBF1  20 0B        JR     NZ,DBFE     nein ? dann Fehler
DBF3  23           INC    HL          Komma übergehen
DBF4  E3           EX     (SP),HL     Eingabe-Zg. retten, PC zurück
DBF5  18 DF        JR     DBD6        nächste Variable behandeln
DBF7  CD 61 DD     CALL   DD61        Spaces, TABs und LFs überlesen
DBFA  B7           OR     A           Zeilenende ?
DBFB  20 01        JR     NZ,DBFE     nein ? dann Fehler
DBFD  37           SCF                sonst CY=1 für fehlerfrei
DBFE  E1           POP    HL
DBFF  E1           POP    HL
DC00  D1           POP    DE
DC01  C9           RET

*********************************** Eingabe auswerten
                                    IN : HL: Eingabezeiger
                                          A: Trennzeichen
                                    OUT: A: Typflag
                                          HL: Eingabezeiger
                                          Eingabewert im FAC
                                          CY=0 bei Fehler
DC02  5F           LD     E,A         Trennzeichen
DC03  CD 45 FF     CALL   FF45        Typflag der Variablen holen
DC06  57           LD     D,A         nach D
```

```
DC07  D5          PUSH   DE          Typflag/Trennzeichen retten
DC08  20 06       JR     NZ,DC10     kein String ? dann numerisch
DC0A  CD 21 DC    CALL   DC21        Eingabestring holen
DC0D  37          SCF                CY=1 für fehlerfrei
DC0E  18 09       JR     DC19        weiter auswerten
DC10  CD C0 C1    CALL   C1C0        Eingabekanalnr. holen
DC13  D4 38 DC    CALL   NC,DC38     Kassette ?
DC16  CD A3 EC    CALL   ECA3        Eingabe nach binär wandeln
DC19  F5          PUSH   AF          Fehlerflag, kein Fehler ?
DC1A  DC 61 DD    CALL   C,DD61      dann Spaces, TABs, LFs überl.
DC1D  F1          POP    AF          Fehlerflag zurück
DC1E  D1          POP    DE          Typflag der Variable
DC1F  7A          LD     A,D         nach A
DC20  C9          RET

*********************************** Eingabestring holen
                                   IN : E: Trennzeichen
                                   OUT: HL: Zeiger auf Stringende
DC21  CD C0 C1    CALL   C1C0        Eingabekanalnr. holen
DC24  38 06       JR     C,DC2C      nicht Kassette ?
DC26  CD 47 DC    CALL   DC47        sonst Eingabestr. von Kassette
DC29  C3 DC F7    JP     F7DC        String auf Stringstack
DC2C  CD 61 DD    CALL   DD61        Spaces, TABs und LFs überlesen
DC2F  FE 22       CP     22          '"' ?
DC31  CA CB F7    JP     Z,F7CB      dann String holen, auf Stack
DC34  7B          LD     A,E         Trennzeichen
DC35  C3 E6 F7    JP     F7E6        String bis Trennzeichen holen

*********************************** numerische Eingabe (von Kassette)
                                   OUT: HL: Zeiger auf Eingabe
DC38  CD 9D DC    CALL   DC9D        Zeichen holen
DC3B  30 05       JR     NC,DC42     Fehler ? dann "EOF met"
DC3D  11 C6 DC    LD     DE,DCC6     Adr. f." ", TAB, LF, CR, ","
DC40  18 2C       JR     DC6E        Eingabe bis Trennzeichen holen

DC42  1E 18       LD     E,18        Nr. für "EOF met"
DC44  C3 94 CA    JP     CA94        Fehler ausgeben

*********************************** Eingabestring (von Kassette)
                                   OUT: HL: Zeiger auf Eingabestring
DC47  CD 9D DC    CALL   DC9D        Zeichen holen
DC4A  30 F6       JR     NC,DC42     Fehler ? dann "EOF met"
DC4C  FE 22       CP     22          '"' ?
DC4E  28 05       JR     Z,DC55      dann String bis '"' holen
DC50  11 CA DC    LD     DE,DCCA     Adr. f. Test auf Komma/CR
DC53  18 19       JR     DC6E        Eingabe bis Trennzeichen holen
DC55  CD A8 DC    CALL   DCA8        Zeichen holen
DC58  11 63 DC    LD     DE,DC63     Adresse für Test auf '"'
DC5B  38 11       JR     C,DC6E      kein EOF ? dann Eingabe holen
DC5D  21 A4 AC    LD     HL,ACA4     sonst Zeiger auf Buffer
DC60  36 00       LD     (HL),00     Null ans Bufferende
DC62  C9          RET

DC63  FE 22       CP     22          Test auf '"'
DC65  C9          RET
```

```
**********************************   Eingabezeile (von Kassette) holen
                                     OUT: HL: Zeiger auf Eingabezeile
DC66  CD A8 DC     CALL    DCA8         Zeichen holen
DC69  30 D7        JR      NC,DC42      Fehler ? dann "EOF met"
DC6B  11 CD DC     LD      DE,DCCD      Adr. für Test auf CR

**********************************   Eingabe bis Trennzeichen holen
                                     IN : A: 1. Eingabezeichen
                                          DE: Adresse der Testroutine
                                              für Trennzeichen
                                          OUT: HL: Zeiger auf Eingabe
DC6E  21 A4 AC     LD      HL,ACA4      Zeiger auf Eingabebuffer
DC71  E5           PUSH    HL           retten
DC72  06 FF        LD      B,FF         max. Bufferlänge
DC74  CD FB FF     CALL    FFFB         Trennzeichen ?
DC77  28 0C        JR      Z,DC85       dann Eingabe-Ende
DC79  77           LD      (HL),A       sonst Zeichen in Buffer
DC7A  23           INC     HL           Bufferzeiger
DC7B  05           DEC     B            restliche Bufferlänge
DC7C  28 05        JR      Z,DC83       Buffer voll ? dann Fehler
DC7E  CD A8 DC     CALL    DCA8         sonst Zeichen holen
DC81  38 F1        JR      C,DC74       kein EOF ? dann weiter
DC83  F6 FF        OR      FF           Flag für Fehler, Z=0
DC85  36 00        LD      (HL),00      Null ans Bufferende
DC87  E1           POP     HL           Zeiger auf Eingabe
DC88  C0           RET     NZ           Fehler ? dann zurück
DC89  FE 0D        CP      0D           CR ?
DC8B  C8           RET     Z            dann zurück
DC8C  FE 22        CP      22           '"' ?
DC8E  C4 D0 DC     CALL    NZ,DCD0      ggf. Test auf Space, TAB, LF
DC91  C0           RET     NZ           nein ? dann zurück
DC92  CD 9D DC     CALL    DC9D         sonst nächstes Zeichen
DC95  D0           RET     NC           EOF ? dann zurück
DC96  CD CA DC     CALL    DCCA         Test auf Komma/CR
DC99  C4 14 C4     CALL    NZ,C414      nein ? dann Zeichen zurück
DC9C  C9           RET

**********************************   Zeichen hol., " ", TAB, LF überl.
                                     OUT: A: Zeichen
                                          CY=0 bei EOF
DC9D  CD A8 DC     CALL    DCA8         Zeichen holen
DCA0  D0           RET     NC           EOF ?
DCA1  CD D0 DC     CALL    DCD0         auf Space, TAB, LF prüfen
DCA4  28 F7        JR      Z,DC9D       ja ? dann nächstes Zeichen
DCA6  37           SCF                  CY=1 für fehlerfrei
DCA7  C9           RET

**********************************   Zeichen holen, CR/LF auswerten
                                     OUT: A: Zeichen
                                          CY=0 bei EOF
DCA8  CD 24 C4     CALL    C424         Zeichen holen
DCAB  D0           RET     NC           EOF ? dann zurück
DCAC  C5           PUSH    BC
DCAD  FE 0D        CP      0D           CR ?
DCAF  06 0A        LD      B,0A         Code für Linefeed
DCB1  28 05        JR      Z,DCB8       CR ? dann auf LF testen
DCB3  B8           CP      B            Linefeed ?
DCB4  20 0D        JR      NZ,DCC3      nein ?
```

```
DCB6  06 OD       LD     B,0D          auf folgendes CR prüfen
DCB8  4F          LD     C,A           Zeichen
DCB9  CD 24 C4    CALL   C424          nächstes Zeichen
DCBC  30 04       JR     NC,DCC2       EOF ?
DCBE  B8          CP     B             sonst auf LF/CR prüfen
DCBF  C4 14 C4    CALL   NZ,C414       nein ? dann Zeichen zurück
DCC2  79          LD     A,C           Zeichen
DCC3  C1          POP    BC
DCC4  37          SCF                  CY=1 für fehlerfrei
DCC5  C9          RET

***********************************  Space, TAB, LF, Komma, CR prüfen
                                     IN : A: Zeichen
                                     OUT: Z=1, wenn Test positiv
DCC6  CD D0 DC    CALL   DCD0            Test auf Space, TAB, LF
DCC9  C8          RET    Z               Test positiv ?
DCCA  FE 2C       CP     2C              "," ?
DCCC  C8          RET    Z
DCCD  FE 0D       CP     0D              CR ?
DCCF  C9          RET

***********************************  auf Space, TAB, LF prüfen
                                     IN : A: Zeichen
                                     OUT: Z=1, wenn Test positiv
DCD0  FE 20       CP     20              Space ?
DCD2  C8          RET    Z
DCD3  FE 09       CP     09              TAB ?
DCD5  C8          RET    Z
DCD6  FE 0A       CP     0A              LF ?
DCD8  C9          RET

***********************************  Basic-Befehl RESTORE
DCD9  28 0A       JR     Z,DCE5          Statementende ?
DCDB  CD E1 CE    CALL   CEE1            Zeilennr. holen, nach DE
DCDE  E5          PUSH   HL              Basic-PC retten
DCDF  CD 9A E7    CALL   E79A            Zeile im Programm suchen
DCE2  2B          DEC    HL              Zeiger auf Zeilenende
DCE3  18 2D       JR     DD12            als DATA-Zeiger setzen
DCE5  E5          PUSH   HL              Basic-PC retten
DCE6  2A 81 AE    LD     HL,(AE81)       Zeiger auf Programmstart
DCE9  18 27       JR     DD12            als DATA-Zeiger setzen

***********************************  Basic-Befehl READ
DCEB  E5          PUSH   HL              Basic-PC retten
DCEC  2A 30 AE    LD     HL,(AE30)       DATA-Zeiger
DCEF  CD 17 DD    CALL   DD17            nächstes DATA-Element suchen
DCF2  E3          EX     (SP),HL
DCF3  CD 86 D6    CALL   D686              Variable holen, Adr. n. DE
DCF6  E3          EX     (SP),HL
DCF7  23          INC    HL              Zeiger auf DATA-Element
DCF8  3E 3A       LD     A,3A            ":" als Trennzeichen
DCFA  CD BC DB    CALL   DBBC            DATA-Element an Var. zuweisen
DCFD  2B          DEC    HL              Zeiger vor nächstes Element
DCFE  28 0B       JR     Z,DD0B          Komma/Zeilenende ?
DD00  2A 2E AE    LD     HL,(AE2E)       sonst DATA-Zeilenadresse
DD03  CD CE DD    CALL   DDCE            als akt. Zeilenadr. f. Fehler
DD06  1E 02       LD     E,02            Nr. für "Syntax error"
DD08  C3 94 CA    JP     CA94            Fehler ausgeben
```

```
DD0B  E3         EX    (SP),HL
DD0C  CD 55 DD   CALL  DD55           folgt Komma im Programm ?
DD0F  E3         EX    (SP),HL
DD10  38 DD      JR    C,DCEF        dann nächste Variable
DD12  22 30 AE   LD    (AE30),HL     DATA-Zeiger wieder speichern
DD15  E1         POP   HL            Basic-PC zurück
DD16  C9         RET

*********************************  nächstes DATA-Element suchen
                                    IN/OUT: HL: DATA-Zeiger
DD17  7E         LD    A,(HL)         nächstes Zeichen
DD18  FE 2C      CP    2C             Komma ?
DD1A  C8         RET   Z              dann fertig
DD1B  CD EF E8   CALL  E8EF           nächstes Statement suchen
DD1E  B7         OR    A
DD1F  20 0E      JR    NZ,DD2F        kein Zeilenende ?
DD21  23         INC   HL
DD22  7E         LD    A,(HL)
DD23  23         INC   HL               nächste Zeilenlänge=0 ?
DD24  B6         OR    (HL)             (Programmende ?)
DD25  23         INC   HL
DD26  1E 04      LD    E,04           Nr. für "DATA exhausted"
DD28  CA 94 CA   JP    Z,CA94         Fehler, wenn Programmende
DD2B  22 2E AE   LD    (AE2E),HL      DATA-Zeilenadresse setzen
DD2E  23         INC   HL             Zeiger auf Zeilennr. hi
DD2F  CD 3F DD   CALL  DD3F           Zeichen aus Zeile
DD32  FE 8C      CP    8C             Token für DATA ?
DD34  20 E5      JR    NZ,DD1B        nein ? dann weiter suchen
DD36  C9         RET

*********************************  Test auf Zeichen nach Aufruf
                                    OUT: A: folgendes Zeichen
                                         CY=0 (immer); Z=1, wenn Statementende
DD37  E3         EX    (SP),HL        PC retten, Aufrufadr. v. Stack
DD38  7E         LD    A,(HL)         Zeichen nach Aufrufbefehl
DD39  23         INC   HL             neue Rücksprungadresse
DD3A  E3         EX    (SP),HL        auf Stack, PC zurück
DD3B  BE         CP    (HL)           mit Zeichen im Programm vergl.
DD3C  C2 C6 DD   JP    NZ,DDC6        ungleich ? dann "Syntax error"

*********************************  nächstes Zeichen holen
                                    OUT: A: Zeichen
                                         CY=0 (immer); Z=1, wenn Statementende
DD3F  23         INC   HL             Basic-PC erhöhen
DD40  7E         LD    A,(HL)         nächstes Zeichen
DD41  FE 20      CP    20             Space ?
DD43  28 FA      JR    Z,DD3F         dann nächstes Zeichen
DD45  FE 01      CP    01             ":" ?
DD47  D0         RET   NC             kein Zeilenende ? dann zurück
DD48  B7         OR    A              Z=1 bei Zeilenende, CY=0
DD49  C9         RET

*********************************  auf Statementende prüfen
DD4A  7E         LD    A,(HL)         Zeichen
DD4B  FE 02      CP    02             ":"/Zeilenende ?
DD4D  D8         RET   C              dann o.k.
DD4E  C3 C6 DD   JP    DDC6           sonst "Syntax error"
```

```
*********************************** Test auf Statementende
                                    OUT: A: Zeichen
                                            CY=1 bei Statementende
DD51  7E          LD    A,(HL)         Zeichen
DD52  FE 02       CP    02             ":"/Zeilenende ?
DD54  C9          RET

*********************************** Test auf Komma
                                    OUT: CY=1, wenn Komma
                                            dann:
                                            A: Zeichen nach Komma
DD55  2B          DEC   HL             vorheriges Zeichen
DD56  CD 3F DD    CALL  DD3F           nächstes Zeichen
DD59  EE 2C       XOR   2C             "," ? (CY=0)
DD5B  C0          RET   NZ             nein ? dann zurück
DD5C  CD 3F DD    CALL  DD3F           sonst nächstes Zeichen
DD5F  37          SCF                  CY=1 für Komma
DD60  C9          RET

*********************************** Spaces, TABs und LFs überlesen
                                    OUT: A: folgendes Zeichen
DD61  7E          LD    A,(HL)         Zeichen
DD62  23          INC   HL
DD63  FE 20       CP    20             Space ?
DD65  28 FA       JR    Z,DD61
DD67  FE 09       CP    09             TAB ?
DD69  28 F6       JR    Z,DD61
DD6B  FE 0A       CP    0A             LF ?
DD6D  28 F2       JR    Z,DD61
DD6F  2B          DEC   HL             sonst Zeiger wieder zurück
DD70  C9          RET

*********************************** Statement nochmals ausführen
DD71  2A 34 AE    LD    HL,(AE34)      Zeiger auf Statement als PC

*********************************** Interpreterschleife
DD74  EB          EX    DE,HL          PC nach DE
DD75  2A 8B B0    LD    HL,(B08B)      Basic-Stackpointer
DD78  22 32 AE    LD    (AE32),HL      als Statementanfangs-SP setzen
DD7B  EB          EX    DE,HL          PC wieder nach HL
DD7C  22 34 AE    LD    (AE34),HL      Statementanfangs-PC setzen
DD7F  CD 21 B9    CALL  B921           KL POLL SYNCHRONOUS
DD82  DC 07 C8    CALL  C,C807         Event bear., wenn Prior. höher
DD85  CD 3F DD    CALL  DD3F           nächstes Zeichen
DD88  C4 AB DD    CALL  NZ,DDAB        Statementende ? sonst Befehl
DD8B  7E          LD    A,(HL)         Zeichen
DD8C  FE 01       CP    01             ":" ?
DD8E  28 E4       JR    Z,DD74         dann nächster Befehl
DD90  30 34       JR    NC,DDC6        kein Zeilenende ? dann Fehler
DD92  23          INC   HL             Zeilenende übergehen
DD93  7E          LD    A,(HL)
DD94  23          INC   HL                nächste Zeilenlänge =0 ?
DD95  B6          OR    (HL)              (Programmende ?)
DD96  23          INC   HL
DD97  28 0F       JR    Z,DDA8         dann Programmende behandeln
DD99  22 36 AE    LD    (AE36),HL      sonst neue Zeilenadr. setzen
DD9C  23          INC   HL             Zeiger vor Zeilentext
DD9D  3A 38 AE    LD    A,(AE38)       Trace-Flag
```

```
DDA0  B7          OR    A           nicht gesetzt ?
DDA1  28 D1       JR    Z,DD74      dann nächster Befehl
DDA3  CD EB DD    CALL  DDEB        Trace-Routine
DDA6  18 CC       JR    DD74        nächster Befehl
DDA8  C3 76 CB    JP    CB76        Programmende behandeln

*********************************  Befehl ausführen
                                   IN : A: Token
DDAB  87          ADD   A           2 Bytes pro Tabelleneintrag
DDAC  D2 4F D6    JP    NC,D64F     Code <$80 ? dann LET/RSX-Code
DDAF  FE B9       CP    B9          Token >$DC ?
DDB1  30 10       JR    NC,DDC3     dann Fehler
DDB3  EB          EX    DE,HL       Basic-PC nach DE
DDB4  C6 01       ADD   01
DDB6  6F          LD    L,A           $DE01, Adresse der Tabelle
DDB7  CE DE       ADC   DE            der Befehlsadressen addieren
DDB9  95          SUB   L
DDBA  67          LD    H,A
DDBB  4E          LD    C,(HL)
DDBC  23          INC   HL            Adresse des Befehls
DDBD  46          LD    B,(HL)        aus Tabelle laden
DDBE  C5          PUSH  BC          Befehlsadresse auf Stack
DDBF  EB          EX    DE,HL       Basic-PC wieder nach HL
DDC0  C3 3F DD    JP    DD3F        nächstes Zeichen, Befehl ausf.

DDC3  CD 07 AC    CALL  AC07        User-Vektor
DDC6  1E 02       LD    E,02        Nr. für "Syntax error"
DDC8  C3 94 CA    JP    CA94        Fehler ausgeben

*********************************  Direkt-Modus einschalten
DDCB  21 00 00    LD    HL,0000     Flag für Direkt-Modus
DDCE  22 36 AE    LD    (AE36),HL   als akt. Zeilenadresse
DDD1  C9          RET

*********************************  Zeilenadresse nach HL holen
DDD2  2A 36 AE    LD    HL,(AE36)   Zeilenadresse
DDD5  C9          RET

*********************************  Zeilennr./Direkt-Modus-Flag holen
                                   OUT: HL: Zeilennummer
                                          Z=1, CY=0, wenn Direkt-Modus
DDD6  2A 36 AE    LD    HL,(AE36)   akt. Zeilenadresse
DDD9  7C          LD    A,H           Kennzeichen für
DDDA  B5          OR    L             Direkt-Modus ? (CY=0!)
DDDB  C8          RET   Z           dann zurück
DDDC  7E          LD    A,(HL)
DDDD  23          INC   HL            Zeilennr. nach HL
DDDE  66          LD    H,(HL)
DDDF  6F          LD    L,A
DDE0  37          SCF               CY=1 für Programm-Modus
DDE1  C9          RET

*********************************  Basic-Befehl TRON
DDE2  3E FF       LD    A,FF        Flag für Trace eingeschaltet
DDE4  18 01       JR    DDE7        setzen
```

```
***********************************   Basic-Befehl TROFF
DDE6  AF          XOR   A              Flag für Trace ausgeschaltet
DDE7  32 38 AE    LD    (AE38),A       Trace-Flag setzen
DDEA  C9          RET

***********************************   Trace-Routine
DDEB  3E 5B       LD    A,5B           eckige Klammer auf
DDED  CD 56 C3    CALL  C356           ausgeben
DDF0  E5          PUSH  HL             Basic-PC retten
DDF1  2A 36 AE    LD    HL,(AE36)      akt. Zeilenadresse
DDF4  7E          LD    A,(HL)
DDF5  23          INC   HL               Zeilennummer laden
DDF6  66          LD    H,(HL)
DDF7  6F          LD    L,A
DDF8  CD 79 EE    CALL  EE79           und ausgeben
DDFB  E1          POP   HL             Basic-PC zurück
DDFC  3E 5D       LD    A,5D           eckige Klammer zu
DDFE  C3 56 C3    JP    C356           ausgeben

***********************************   Adressen der Basic-Befehle
DE01  71 C9 DF C0 21 C2 BA F1          AFTER, AUTO, BORDER, CALL
DE09  46 D2 3C EA 32 C1 B5 C4          CAT, CHAIN, CLEAR, CLG
DE11  98 D2 A1 D2 5A C2 C0 CB          CLOSEIN, CLOSEOUT, CLS, CONT
DE19  EF E8 17 D1 18 D6 1C D6          DATA, DEF, DEFINT, DEFREAL
DE21  14 D6 E7 D4 28 E7 7D D6          DEFSTR, DEG, DELETE, DIM
DE29  C6 C4 CB C4 52 C0 F3 E8          DRAW, DRAWR, EDIT, ELSE
DE31  65 CB 85 D3 4E D3 C0 D9          END, ENT, ENV, ERASE
DE39  8F CA 79 C9 29 C5 ED C6          ERROR, EVERY, FOR, GOSUB
DE41  E8 C6 C7 C6 2A C2 2B DB          GOTO, IF, INK, INPUT
DE49  39 D4 54 D6 F8 DA F7 E0          KEY, LET, LINE, LIST
DE51  F6 E9 D2 C2 EF F4 A6 EA          LOAD, LOCATE, MEMORY, MERGE
DE59  93 F9 4F C2 05 C5 0A C5          MID$, MODE, MOVE, MOVER
DE61  FB C5 2B C1 E3 C7 CB C8          NEXT, NEW, ON, ON BREAK
DE69  F8 CB 40 C9 5F D2 56 D2          ON ERROR GOTO 0, ON SQ,
                                       OPENIN, OPENOUT
DE71  8C C4 77 F1 0A C2 12 C2          ORIGIN, OUT, PAPER, PEN
DE79  D0 C4 D5 C4 5F F1 FD F1          PLOT, PLOTR, POKE, PRINT
DE81  F3 E8 EB D4 59 D5 EB DC          ', RAD, RANDOMIZE, READ
DE89  1E D3 F3 E8 DF E7 D9 DC          RELEASE, REM, RENUM, RESTORE
DE91  03 CC 0F C7 BD E9 09 EC          RESUME, RETURN, RUN, SAVE
DE99  C0 D2 94 D4 5A CB 9D F6          SOUND, SPEED, STOP, SYMBOL
DEA1  19 C3 20 C3 E6 DD E2 DD          TAG, TAGOFF, TROFF, TRON
DEA9  7D F1 76 C7 47 C7 E3 C3          WAIT, WEND, WHILE, WIDTH
DEB1  E1 C2 7B F4 F6 F1 E1 C8          WINDOW, WRITE, ZONE, DI
DEB9  E7 C8                            EI

***********************************   Zeile tokenisieren
                                      IN : HL: Zeiger auf Eingabezeile
                                      OUT: HL: Zeiger auf token. Zeile
                                           BC: Länge der tokenis. Zeile
DEBB  D5          PUSH  DE
DEBC  EB          EX    DE,HL            Start des freien RAMs
DEBD  2A 7F AE    LD    HL,(AE7F)        als Buffer für tokenisierte
DEC0  EB          EX    DE,HL            Zeiger auf Zeile nach DE
DEC1  D5          PUSH  DE               Zeiger auf Buffer retten
DEC2  AF          XOR   A              Flag für Variable/Zeilennr.
DEC3  32 39 AE    LD    (AE39),A       löschen
DEC6  01 2C 01    LD    BC,012C        max. Bufferlänge
```

```
DEC9  CD E1 DE    CALL   DEE1          nächstes Item tokenisieren
DECC  7E          LD     A,(HL)        Zeichen
DECD  B7          OR     A             Zeilenende ?
DECE  20 F9       JR     NZ,DEC9       nein ? d. weiter tokenisieren
DED0  3E 2D       LD     A,2D            $012D (max. Länge)
DED2  91          SUB    C               - restliche Länge
DED3  4F          LD     C,A             ergibt
DED4  3E 01       LD     A,01            Länge der
DED6  98          SBC    B               tokenisierten
DED7  47          LD     B,A             Zeile, nach BC
DED8  AF          XOR    A             Null
DED9  12          LD     (DE),A        ans Zeilenende
DEDA  13          INC    DE
DEDB  12          LD     (DE),A          zwei Nullen als Kennzeichen
DEDC  13          INC    DE              für Programmende
DEDD  12          LD     (DE),A
DEDE  E1          POP    HL            Zeiger auf tokenisierte Zeile
DEDF  D1          POP    DE
DEE0  C9          RET

*********************************** ein Item tokenisieren
DEE1  CD 10 AC    CALL   AC10          User-Vektor
DEE4  7E          LD     A,(HL)        Zeichen aus Eingabe
DEE5  B7          OR     A             Zeilenende ?
DEE6  C8          RET    Z             dann zurück
DEE7  CD 71 FF    CALL   FF71          Buchstabe ?
DEEA  38 1D       JR     C,DF09        dann auswerten
DEEC  CD 7F FF    CALL   FF7F          Ziffer oder Dezimalpunkt ?
DEEF  DA FF DF    JP     C,DFFF        dann Dezimalzahl auswerten
DEF2  FE 26       CP     26            "&" ?
DEF4  CA 5A E0    JP     Z,E05A        dann Hex- oder Binärzahl
DEF7  23          INC    HL            Zeiger auf nächstes Zeichen
DEF8  FE 80       CP     80            Sonderzeichen >=$80 ?
DEFA  D0          RET    NC            dann zurück
DEFB  FE 20       CP     20            Space ?
DEFD  C2 80 E0    JP     NZ,E080       nein ? dann weiter prüfen
DF00  3A 00 AC    LD     A,(AC00)      Flag für Space-Unterdrückung
DF03  B7          OR     A             gesetzt ?
DF04  C0          RET    NZ            dann zurück
DF05  3E 20       LD     A,20          sonst Space
DF07  18 1C       JR     DF25          in Buffer schreiben

*********************************** Buchstaben auswerten
DF09  CD 4E DF    CALL   DF4E          Keyword/Variable tokenisieren
DF0C  D8          RET    C             kein Befehls-Token ? d. zurück
DF0D  FE C5       CP     C5            Token für REM ?
DF0F  CA ED E0    JP     Z,E0ED        dann restl. Zeile übernehmen
DF12  E5          PUSH   HL            Eingabezeiger retten
DF13  21 30 DF    LD     HL,DF30       Zeiger auf Tabelle
DF16  CD AA FF    CALL   FFAA          Token in Tabelle enthalten ?
DF19  E1          POP    HL            Eingabezeiger
DF1A  38 19       JR     C,DF35        dann bis Statementende übern.
DF1C  F5          PUSH   AF            Token retten
DF1D  FE 97       CP     97            Token für ELSE ?
DF1F  3E 01       LD     A,01          Token für ":"
DF21  CC 25 DF    CALL   Z,DF25        dann mit ":" abspeichern
DF24  F1          POP    AF            Token zurück
```

```
*********************************** Zeichen in Token-Buffer
                                    IN : A: Zeichen
                                        (DE: Bufferzeiger
                                         BC: restl. Bufferlänge)
DF25  12          LD    (DE),A         Zeichen abspeichern
DF26  13          INC   DE             Bufferzeiger erhöhen
DF27  0B          DEC   BC             restliche Bufferlänge
DF28  79          LD    A,C
DF29  B0          OR    B                noch Platz im Buffer ?
DF2A  C0          RET   NZ             dann o.k.
DF2B  1E 17       LD    E,17           Nr. für "Line too long"
DF2D  C3 94 CA    JP    CA94           Fehler ausgeben

*********************************** Tabelle der Tokens mit Sonderteil
DF30  8C                               DATA
DF31  8E                               DEFINT
DF32  90                               DEFSTR
DF33  8F                               DEFREAL
DF34  00                               Tabellenende

*********************************** Zl. bis Statementende übernehmen
DF35  CD 25 DF    CALL  DF25           Zeichen in Buffer speichern
DF38  7E          LD    A,(HL)         Zeichen aus Eingabe
DF39  B7          OR    A              Zeilenende ?
DF3A  C8          RET   Z              dann zur
DF3B  FE 3A       CP    3A             ":" ?
DF3D  28 0A       JR    Z,DF49         dann Flags löschen, zurück
DF3F  23          INC   HL             Eingabezeiger
DF40  FE 22       CP    22             '"' ?
DF42  20 F1       JR    NZ,DF35        nein ? dann weiter übertragen
DF44  CD BF E0    CALL  E0BF           String übertragen
DF47  18 EF       JR    DF38           weiter übertragen
DF49  AF          XOR   A              Flag für Variable/Zeilennr.
DF4A  32 39 AE    LD    (AE39),A       löschen
DF4D  C9          RET

*********************************** Keyword/Variable tokenisieren
                                    OUT: CY=0 für Befehlstoken
DF4E  C5          PUSH  BC             restliche Bufferlänge,
DF4F  D5          PUSH  DE             Bufferzeiger
DF50  E5          PUSH  HL             und Eingabezeiger retten
DF51  CD 16 AC    CALL  AC16           User-Vektor
DF54  7E          LD    A,(HL)         Zeichen
DF55  23          INC   HL             Eingabezeiger
DF56  CD 8A FF    CALL  FF8A           auf Großschrift forcieren
DF59  CD DD E2    CALL  E2DD           entspr. Keyword-Tabellenadr.
DF5C  CD 27 E3    CALL  E327           Eingabe in Keyword-Tabelle s.
DF5F  30 28       JR    NC,DF89        nicht gefunden ? d. Var.-Name
DF61  79          LD    A,C            letztes Zeichen des Keywords
DF62  E6 7F       AND   7F
DF64  CD 7B FF    CALL  FF7B           Ziffer, Buchstabe oder "." ?
DF67  30 0B       JR    NC,DF74        nein ? dann Keyword gefunden
DF69  1A          LD    A,(DE)         Token laden
DF6A  FE E4       CP    E4             Token für FN ?
DF6C  28 06       JR    Z,DF74         dann Keyword gefunden
DF6E  7E          LD    A,(HL)         Zeichen nach Keyword
DF6F  CD 7B FF    CALL  FF7B           Ziffer, Buchstabe oder "." ?
DF72  38 15       JR    C,DF89         dann nicht zu Ende, Variable
```

```
DF74  F1          POP   AF          Eingabezeiger löschen
DF75  1A          LD    A,(DE)      Token laden
DF76  B7          OR    A
DF77  FA C8 DF    JP    M,DFC8      Befehls-Token ?
DF7A  D1          POP   DE          Bufferzeiger
DF7B  C1          POP   BC          und restl. Länge zurück
DF7C  F5          PUSH  AF          Token retten
DF7D  3E FF       LD    A,FF        Kennz. für Funktion
DF7F  CD 25 DF    CALL  DF25        in Buffer
DF82  F1          POP   AF          Funktions-Token
DF83  CD 25 DF    CALL  DF25        in Buffer
DF86  AF          XOR   A           Flag für Variable/Zeilennr.
DF87  18 3A       JR    DFC3        löschen

*********************************  Variablennamen auswerten
DF89  E1          POP   HL          Eingabezeiger auf Namen
DF8A  D1          POP   DE          Token-Buffer-Zeiger
DF8B  C1          POP   BC          restliche Bufferlänge
DF8C  E5          PUSH  HL          Eingabezeiger
DF8D  2B          DEC   HL
DF8E  23          INC   HL          nächstes Zeichen
DF8F  7E          LD    A,(HL)      laden
DF90  CD 7B FF    CALL  FF7B        Buchstabe, Ziffer, "." ?
DF93  38 F9       JR    C,DF8E      dann nächstes Zeichen
DF95  CD EA DF    CALL  DFEA        Variablentyp prüfen
DF98  38 04       JR    C,DF9E      Kennz. ("%", "$", "!") ?
DF9A  3E 0D       LD    A,0D        sonst Token f. umarkierte Var.
DF9C  18 06       JR    DFA4        abspeichern
DF9E  23          INC   HL          Eingabezeiger
DF9F  FE 05       CP    05          REAL-Variable ?
DFA1  20 01       JR    NZ,DFA4     sonst abspeichern
DFA3  3D          DEC   A           4, Token für REAL-Variable
DFA4  CD 25 DF    CALL  DF25        in Token-Buffer
DFA7  AF          XOR   A
DFA8  CD 25 DF    CALL  DF25          2 Nullen für Offset
DFAB  AF          XOR   A             in Token-Buffer
DFAC  CD 25 DF    CALL  DF25
DFAF  E3          EX    (SP),HL     Eing.-Zg. retten, alter zurück
DFB0  7E          LD    A,(HL)      Zeichen aus Variablenname
DFB1  CD 7B FF    CALL  FF7B        Buchstabe, Ziffer oder "." ?
DFB4  30 07       JR    NC,DFBD     nein ? dann Name zu Ende
DFB6  7E          LD    A,(HL)      Zeichen
DFB7  CD 25 DF    CALL  DF25        in Token-Buffer speichern
DFBA  23          INC   HL          Eingabezeiger
DFBB  18 F3       JR    DFB0        nächstes Zeichen
DFBD  CD DF E0    CALL  E0DF        Endmarkierung setzen
DFC0  E1          POP   HL          Zeiger nach Variable
DFC1  3E FF       LD    A,FF        Kennz. für Variable
DFC3  32 39 AE    LD    (AE39),A    setzen
DFC6  37          SCF               CY=1 für kein Befehlstoken
DFC7  C9          RET

*********************************  Befehlstoken behandeln
                                   IN/OUT : A: Token
                                   CY=0, da Befehlstoken
DFC8  E5          PUSH  HL          Eingabezeiger
DFC9  4F          LD    C,A         Token
DFCA  21 DC DF    LD    HL,DFDC     Tabellenadr. f. Zeilennr.-Bef.
```

```
DFCD  CD AA FF     CALL   FFAA          Token in Tabelle suchen
DFD0  9F           SBC    A             A=$FF, wenn gefunden, sonst 0
DFD1  E6 01        AND    01            A=1, wenn gefunden, sonst 0
DFD3  32 39 AE     LD     (AE39),A      Flag für Zeilennr. setzen
DFD6  79           LD     A,C           Token
DFD7  E1           POP    HL            Eingabezeiger
DFD8  D1           POP    DE            Token-Buffer-Zeiger
DFD9  C1           POP    BC            und restl. Bufferlänge
DFDA  B7           OR     A             CY=0 für Befehlstoken
DFDB  C9           RET

***********************************   Tabelle der Tokens mit Zeilennr.
DFDC  C7 81 C6 92                         RESTORE, AUTO, RENUM, DELETE
DFE0  96 C8 E3 97                         EDIT, RESUME, ERL, ELSE
DFE4  CA A7 A0 EB                         RUN, LIST, GOTO, THEN
DFE8  9F 00                               GOSUB, Tabellenende

***********************************   Variablentyp feststellen
                                      IN : A: Zeichen
                                      OUT: CY=1 für "$", "%", "!"
                                              A: Typ
DFEA  FE 26        CP     26            größer "%" ?
DFEC  D0           RET    NC            dann zurück
DFED  FE 21        CP     21            kleiner "!" ?
DFEF  3F           CCF
DFF0  D0           RET    NC            dann zurück
DFF1  FE 22        CP     22            '"' ?
DFF3  C8           RET    Z             dann zurück
DFF4  FE 23        CP     23            "#" ?
DFF6  C8           RET    Z             dann zurück
DFF7  EE 27        XOR    27              Variablentyp (2,3,5)
DFF9  FE 04        CP     04              generieren
DFFB  CE FF        ADC    FF
DFFD  37           SCF                  CY=1 für Variablenende
DFFE  C9           RET

***********************************   Dezimalzahl tokenisieren
DFFF  3A 39 AE     LD     A,(AE39)        Flag für Variable/Zeilennr.
E002  B7           OR     A             Flag für Zeilennr. ?
E003  28 15        JR     Z,E01A        nein ? dann Integer/REAL-Zahl
E005  7E           LD     A,(HL)        nächstes Zeichen
E006  23           INC    HL            Eingabezeiger
E007  FA 25 DF     JP     M,DF25        Zeichen >=$80 ? dann speichern
E00A  FE 2E        CP     2E            "." ?
E00C  CA 25 DF     JP     Z,DF25        dann so speichern
E00F  2B           DEC    HL            Zeiger wieder auf 1. Ziffer
E010  D5           PUSH   DE            Bufferzeiger retten
E011  CD 04 EE     CALL   EE04          String n. Zahl im FAC wandeln
E014  30 34        JR     NC,E04A       Fehler ? d. Zahl so speichern
E016  3E 1E        LD     A,1E          Token für Zeilenummer
E018  18 4F        JR     E069          mit Zeilennr. im FAC speichern
E01A  D5           PUSH   DE            Bufferzeiger und
E01B  C5           PUSH   BC            restl. Bufferlänge retten
E01C  CD BE EC     CALL   ECBE          String in pos. Binärzahl
E01F  C1           POP    BC            restl. Bufferlänge
E020  30 28        JR     NC,E04A       Fehler ? d. Zahl so speichern
E022  CD 27 FF     CALL   FF27          Typflag des FAC holen
E025  3E 1F        LD     A,1F          Token für REAL-Zahl
```

```
E027  30 40     JR    NC,E069     REAL-Zahl im FAC ?
E029  EB        EX    DE,HL
E02A  2A C2 B0  LD    HL,(B0C2)     Integerwert aus FAC nach DE
E02D  EB        EX    DE,HL
E02E  7A        LD    A,D         Hi-Byte
E02F  B7        OR    A
E030  3E 1A     LD    A,1A        Token für dez. Integerwert
E032  20 35     JR    NZ,E069     Hi-Byte <>0 ? dann 2-Byte-Wert
E034  E3        EX    (SP),HL     Eing.-Zg. retten, Bufferzeiger
E035  EB        EX    DE,HL       nach DE, Integerwert nach HL
E036  7D        LD    A,L         Lo-Byte
E037  FE 0A     CP    0A          >=10 ?
E039  30 04     JR    NC,E03F     dann Ein-Byte-Wert speichern
E03B  C6 0E     ADD   0E          sonst Token generieren
E03D  18 06     JR    E045        Wert im Token enthalten sp.
E03F  3E 19     LD    A,19        Token für Ein-Byte-Wert
E041  CD 25 DF  CALL  DF25        in Token-Buffer
E044  7D        LD    A,L         Byte
E045  CD 25 DF  CALL  DF25        in Buffer
E048  E1        POP   HL          Eingabezeiger zurück
E049  C9        RET

*********************************** Eingabe bis DE übernehmen
                                  IN : DE: Endzeiger
                                          HL: Eingabezeiger
                                          TOS: Token-Buffer-Zeiger
E04A  7E        LD    A,(HL)        Zeichen aus Eingabe
E04B  23        INC   HL            Eingabezeiger
E04C  E3        EX    (SP),HL       retten, Bufferzeiger zurück
E04D  EB        EX    DE,HL         nach DE, Endzeiger nach HL
E04E  CD 25 DF  CALL  DF25          Zeichen in Buffer
E051  EB        EX    DE,HL         Endzeiger n. DE, Bufferzeiger
E052  E3        EX    (SP),HL       auf Stack, Eingabezg. n. HL
E053  CD B8 FF  CALL  FFB8          mit Endzeiger vergleichen
E056  20 F2     JR    NZ,E04A       Ende noch nicht erreicht ?
E058  D1        POP   DE            Bufferzeiger zurück
E059  C9        RET

*********************************** Hex/Binärzahl tokenisieren
E05A  D5        PUSH  DE            Token-Buffer-Zeiger
E05B  C5        PUSH  BC            und restl. Bufferlänge
E05C  CD BE EC  CALL  ECBE          String in pos. Binärzahl
E05F  C1        POP   BC            restl. Länge
E060  30 E8     JR    NC,E04A       Überlauf ? dann so speichern
E062  FE 02     CP    02            Basis =2 ?
E064  3E 1B     LD    A,1B          Token für Binärzahl
E066  28 01     JR    Z,E069        Binärzahl ?
E068  3C        INC   A             sonst Token für Hex-Zahl
E069  D1        POP   DE            Token-Buffer-Zeiger
E06A  CD 25 DF  CALL  DF25          Token in Buffer
E06D  E5        PUSH  HL            Eingabezeiger
E06E  21 C2 B0  LD    HL,B0C2       Zeiger auf FAC
E071  CD 23 FF  CALL  FF23          Typflag (Länge) des FAC holen
E074  F5        PUSH  AF            als Zähler für Bytes
E075  7E        LD    A,(HL)        Byte aus FAC
E076  23        INC   HL
E077  CD 25 DF  CALL  DF25          in Buffer übertragen
E07A  F1        POP   AF
```

```
E07B  3D          DEC    A           Zähler
E07C  20 F6       JR     NZ,E074     weitere Bytes ?
E07E  E1          POP    HL          Eingabezeiger zurück
E07F  C9          RET

********************************** Sonderzeichen auswerten
E080  FE 22       CP     22            '"' ?
E082  28 3B       JR     Z,E0BF        dann String übertragen
E084  FE 7C       CP     7C            RSX-Kennzeichen ?
E086  28 45       JR     Z,E0CD        dann auswerten
E088  C5          PUSH   BC            restl. Bufferlänge
E089  D5          PUSH   DE            und Bufferzeiger retten
E08A  EE 3F       XOR    3F            "?" ?
E08C  06 BF       LD     B,BF          Token für PRINT
E08E  28 16       JR     Z,E0A6        ggf. "?" ersetzen
E090  2B          DEC    HL            Zeiger für Suche korrigieren
E091  11 4B E6    LD     DE,E64B       Tab. d. Keywords ohne Buchst.
E094  CD 27 E3    CALL   E327          Eingabe in Tabelle suchen
E097  1A          LD     A,(DE)        Token laden
E098  38 08       JR     C,E0A2        in Tabelle gefunden ?
E09A  7E          LD     A,(HL)        sonst Zeichen aus Eingabe
E09B  FE 20       CP     20
E09D  30 02       JR     NC,E0A1       kein Steuerzeichen ?
E09F  3E 20       LD     A,20          sonst durch Space ersetzen
E0A1  23          INC    HL            Eingabezeiger
E0A2  47          LD     B,A           Token/Zeichen
E0A3  CD B3 E0    CALL   E0B3          Flag für Var./Zeilennr. prüfen
E0A6  32 39 AE    LD     (AE39),A      und neu setzen
E0A9  78          LD     A,B           Token/Zeichen
E0AA  D1          POP    DE            Bufferzeiger und
E0AB  C1          POP    BC            restl. Bufferlänge zurück
E0AC  FE C0       CP     C0            Token für "'" ?
E0AE  28 36       JR     Z,E0E6        dann auswerten
E0B0  C3 25 DF    JP     DF25          sonst in Buffer speichern

********************************** Flag f. Variable/Zeilennr. prüfen
                                     IN : A: Zeichen/Token
                                     OUT: A: neues Flag
E0B3  3D          DEC    A             Token für ":" ?
E0B4  C8          RET    Z             dann Flags löschen
E0B5  EE 22       XOR    22            '"' ?
E0B7  C8          RET    Z             dann Flags löschen
E0B8  3A 39 AE    LD     A,(AE39)      alte Flags
E0BB  3C          INC    A             Flag für Variablenname ?
E0BC  C8          RET    Z             dann löschen
E0BD  3D          DEC    A             sonst Flag erhalten
E0BE  C9          RET

********************************** String in Buffer übernehmen
                                     IN : A: Zeichen vor String
E0BF  CD 25 DF    CALL   DF25          Zeichen in Buffer speichern
E0C2  7E          LD     A,(HL)        nächstes Zeichen
E0C3  B7          OR     A             Zeilenende ?
E0C4  C8          RET    Z             dann zurück
E0C5  23          INC    HL            Eingabezeiger
E0C6  FE 22       CP     22            '"' ?
E0C8  20 F5       JR     NZ,E0BF       nein ? dann weiter übernehemen
E0CA  C3 25 DF    JP     DF25          letztes Zeichen in Buffer
```

```
*********************************  RSX-Code auswerten
E0CD  CD 25 DF     CALL   DF25        RSX-Token in Buffer
E0D0  AF           XOR    A           Flag für Variable/Zeilennr.
E0D1  32 39 AE     LD     (AE39),A    löschen
E0D4  CD 25 DF     CALL   DF25        Null in Buffer
E0D7  7E           LD     A,(HL)      Zeichen aus Eingabe
E0D8  23           INC    HL          Eingabezeiger
E0D9  CD 7B FF     CALL   FF7B        Buchstabe, Ziffer oder "." ?
E0DC  38 F6        JR     C,E0D4      dann weiter übertragen
E0DE  2B           DEC    HL          Zeiger auf letztes Byte

*********************************  Kennzeichen für Namen setzen
E0DF  1B           DEC    DE          Token-Buffer-Zeiger
E0E0  1A           LD     A,(DE)      letztes Zeichen aus Buffer
E0E1  F6 80        OR     80          Kennzeichen setzen
E0E3  12           LD     (DE),A
E0E4  13           INC    DE          Buffer-Zeiger wieder zurück
E0E5  C9           RET

*********************************  "'" auswerten
E0E6  3E 01        LD     A,01        Token für ":"
E0E8  CD 25 DF     CALL   DF25        in Buffer
E0EB  3E C0        LD     A,C0        Token für "'"
E0ED  CD 25 DF     CALL   DF25        in Buffer

*********************************  restliche Zeile übernehmen
E0F0  7E           LD     A,(HL)      Zeichen
E0F1  23           INC    HL
E0F2  B7           OR     A           Zeilenende ?
E0F3  20 F8        JR     NZ,E0ED     nein ? dann weiter übernehmen
E0F5  2B           DEC    HL          Zeiger auf Zeilenende
E0F6  C9           RET

*********************************  Basic-Befehl LIST
E0F7  CD B0 CE     CALL   CEB0        Zeilennummernbereich holen
E0FA  C5           PUSH   BC          Start-
E0FB  D5           PUSH   DE          und Endzeilennr. retten
E0FC  CD C6 C1     CALL   C1C6        opt. Filenr. als Streamnr.
E0FF  CD 4A DD     CALL   DD4A        auf Statementende prüfen
E102  CD CB DD     CALL   DDCB        Direkt-Modus einschalten
E105  D1           POP    DE          End-
E106  C1           POP    BC          und Startzeilennr. zurück
E107  CD 0D E1     CALL   E10D        Bereich listen
E10A  C3 64 C0     JP     C064        zur Eingabeschleife

*********************************  Programmbereich listen
                                   IN : BC: Startzeilennr.
                                        DE: Endzeilennr.
E10D  D5           PUSH   DE          Endzeilennr. retten
E10E  50           LD     D,B           Startzeilennr.
E10F  59           LD     E,C           nach DE
E110  CD A3 E7     CALL   E7A3        Zeile im Programm suchen
E113  D1           POP    DE          Endzeilennr.
E114  4E           LD     C,(HL)
E115  23           INC    HL            Zeilenlänge aus Zeile
E116  46           LD     B,(HL)        laden, nach BC
E117  2B           DEC    HL
E118  78           LD     A,B
```

```
E119  B1          OR     C             Zeilenlänge=0 ?
E11A  C8          RET    Z           dann Programmende, fertig
E11B  CD 3C C4    CALL   C43C        Test auf Break-(ESC-)Taste
E11E  E5          PUSH   HL          Zeiger auf Zeile retten
E11F  09          ADD    HL,BC       Länge add., Zg. nächste Zeile
E120  E3          EX     (SP),HL     retten, Zeiger auf Zeile zur.
E121  D5          PUSH   DE          Endzeilennr.
E122  E5          PUSH   HL          und Zeiger auf Zeile retten
E123  23          INC    HL            Zeilenlänge
E124  23          INC    HL            übergehen
E125  5E          LD     E,(HL)
E126  23          INC    HL            Zeilennummer laden
E127  56          LD     D,(HL)
E128  E1          POP    HL            zweitobersten Stackeintrag
E129  E3          EX     (SP),HL       (Endzeilennr.) nach HL
E12A  CD B8 FF    CALL   FFB8        mit akt. Zeilennr. vergleichen
E12D  E3          EX     (SP),HL     Endznr. retten, Zg. Zeile zur.
E12E  38 12       JR     C,E142      akt. Zeilennr. größer ?
E130  CD 63 E1    CALL   E163        Zeile nach ASCII wandeln
E133  CD 45 E1    CALL   E145        Zeichen ausgeben
E136  23          INC    HL
E137  7E          LD     A,(HL)      nächstes Zeichen
E138  B7          OR     A           Zeilenende ?
E139  20 F8       JR     NZ,E133     nein ? dann weiter ausgeben
E13B  CD 4E C3    CALL   C34E        Linefeed ausgeben
E13E  D1          POP    DE          Endzeilennr.
E13F  E1          POP    HL          Zeiger auf Zeile
E140  18 D2       JR     E114        nächste Zeile ausgeben
E142  E1          POP    HL          Endzeilennr. und
E143  E1          POP    HL          Zeiger auf Zeile vom Stack
E144  C9          RET

*********************************** Zeichen für LIST ausgeben
                                    IN : HL: Zeiger auf Zeichen
E145  CD BA C1    CALL   C1BA        aktuelle Streamnr. holen
E148  38 0B       JR     C,E155      Bildschirm ?
E14A  7E          LD     A,(HL)      sonst Zeichen
E14B  CD 6E C3    CALL   C36E        ausgeben
E14E  FE 0A       CP     0A          Linefeed ?
E150  C0          RET    NZ          nein ?
E151  3E 0D       LD     A,0D        sonst CR
E153  18 0B       JR     E160        ausgeben
E155  7E          LD     A,(HL)      Zeichen
E156  FE 20       CP     20          Steuerzeichen ?
E158  30 06       JR     NC,E160     nein ? dann so ausgeben
E15A  3E 01       LD     A,01          sonst Zeichen direkt
E15C  CD 6E C3    CALL   C36E          auf Bildschirm ausgeben
E15F  7E          LD     A,(HL)      Zeichen
E160  C3 6E C3    JP     C36E        ausgeben

*********************************** Basic-Zeile nach ASCII wandeln
                                    IN/OUT: HL: Zeiger auf Zeile
E163  D5          PUSH   DE
E164  01 A4 AC    LD     BC,ACA4       Zeiger auf Buffer
E167  C5          PUSH   BC          retten
E168  23          INC    HL            Zeilenlänge
E169  23          INC    HL            übergehen
E16A  5E          LD     E,(HL)
```

```
E16B  23          INC   HL            Zeilennummer
E16C  56          LD    D,(HL)        laden, nach DE
E16D  23          INC   HL
E16E  E5          PUSH  HL           Zeiger retten
E16F  EB          EX    DE,HL        Zeilennr. nach DE
E170  CD 0D FF    CALL  FF0D         und in FAC eintragen
E173  CD 82 EE    CALL  EE82         nach ASCII wandeln
E176  11 00 00    LD    DE,0000      max. Länge/Flag für Space
E179  7E          LD    A,(HL)       Zeichen aus Zeilennr.-Buffer
E17A  23          INC   HL
E17B  B7          OR    A            Zeilenende ?
E17C  28 05       JR    Z,E183       dann Nr. fertig übertragen
E17E  CD FE E1    CALL  E1FE         Zeichen in Buffer
E181  18 F6       JR    E179         nächstes Zeichen
E183  3E 20       LD    A,20         Space
E185  CD FE E1    CALL  E1FE         in Buffer
E188  E1          POP   HL           Zeiger auf Zeilentext
E189  7E          LD    A,(HL)       Zeichen
E18A  B7          OR    A            Zeilenende ?
E18B  28 05       JR    Z,E192       dann fertig
E18D  CD 96 E1    CALL  E196         Item nach ASCII wandeln
E190  18 F7       JR    E189         nächstes Item
E192  02          LD    (BC),A       Null ans Bufferende
E193  E1          POP   HL           Zeiger auf Zeile
E194  D1          POP   DE
E195  C9          RET

*********************************** Item nach ASCII wandeln
E196  CD 13 AC    CALL  AC13         User-Vektor
E199  FA 20 E2    JP    M,E220       Token für Keyword ?
E19C  FE 02       CP    02
E19E  38 1D       JR    C,E1BD       Statementende ?
E1A0  FE 05       CP    05
E1A2  38 43       JR    C,E1E7       Variable ohne Kennzeichen ?
E1A4  FE 0B       CP    0B
E1A6  38 22       JR    C,E1CA       (??)
E1A8  FE 0E       CP    0E
E1AA  38 3B       JR    C,E1E7       Variable mit Kennzeichen ?
E1AC  FE 20       CP    20
E1AE  38 2E       JR    C,E1DE       Konstante ?
E1B0  FE 7C       CP    7C
E1B2  28 51       JR    Z,E205       RSX-Token ?
E1B4  CD EA DF    CALL  DFEA         Test auf "$", "%", "!"
E1B7  DC 1A E2    CALL  C,E21A       Variablenende ? d. ggf. Space
E1BA  7E          LD    A,(HL)       Zeichen
E1BB  18 0D       JR    E1CA         in Buffer
E1BD  23          INC   HL
E1BE  7E          LD    A,(HL)       Zeichen nach Statementende
E1BF  FE C0       CP    C0           Token für "'" ?
E1C1  28 5D       JR    Z,E220       dann auswerten
E1C3  FE 97       CP    97           Token für ELSE ?
E1C5  28 59       JR    Z,E220       dann auswerten
E1C7  2B          DEC   HL           Zeiger wieder zurück
E1C8  3E 3A       LD    A,3A         ":"
E1CA  1E 00       LD    E,00         Flag für kein folgendes Space
E1CC  FE 22       CP    22           '"' ?
E1CE  20 0B       JR    NZ,E1DB      nein ? dann Code speichern
E1D0  CD FE E1    CALL  E1FE         Zeichen in Buffer
```

```
E1D3  23          INC   HL
E1D4  7E          LD    A,(HL)      nächstes Zeichen
E1D5  B7          OR    A           Zeilenende ?
E1D6  C8          RET   Z           dann zurück
E1D7  FE 22       CP    22          '"' ?
E1D9  20 F5       JR    NZ,E1D0     nein ? dann weiter übernehmen
E1DB  23          INC   HL          Zeiger auf nächstes Zeichen
E1DC  18 20       JR    E1FE        Zeichen in Buffer

********************************** Konstante auswerten
E1DE  CD 1A E2    CALL  E21A          ggf. Space in Buffer
E1E1  CD 53 E2    CALL  E253          Konstante nach ASCII wandeln
E1E4  1E 01       LD    E,01          Flag für nachfolgendes Space
E1E6  C9          RET

********************************** Variable auswerten
E1E7  CD 1A E2    CALL  E21A          ggf. Space in Buffer
E1EA  7E          LD    A,(HL)        Variablen-Token
E1EB  F5          PUSH  AF            retten
E1EC  23          INC   HL
E1ED  23          INC   HL              Offset übergehen,
E1EE  23          INC   HL              Zeiger auf Namen
E1EF  CD 0F E2    CALL  E20F          Namen übertragen
E1F2  F1          POP   AF            Variablen-Token
E1F3  1E 01       LD    E,01          Flag für folgendes Space
E1F5  FE 0B       CP    0B            Variable ohne Kennzeichen ?
E1F7  D0          RET   NC            dann fertig
E1F8  1E 00       LD    E,00          Flag für kein Space
E1FA  EE 27       XOR   27              Kennzeichen ("$", "%"
E1FC  E6 FD       AND   FD              oder "!" generieren)

********************************** Zeichen in LIST-Buffer
                                    IN : A: Zeichen
E1FE  02          LD    (BC),A        Zeichen speichern
E1FF  03          INC   BC            Bufferzeiger
E200  15          DEC   D             restl. Bufferlänge
E201  C0          RET   NZ            dann o.k.
E202  0B          DEC   BC            sonst Zeiger
E203  14          INC   D             und Länge wieder zurück
E204  C9          RET

********************************** RSX-Code auswerten
E205  1E 01       LD    E,01          Flag für folgendes Space
E207  CD FE E1    CALL  E1FE          RSX-Code in Buffer
E20A  23          INC   HL
E20B  7E          LD    A,(HL)        nächstes Zeichen
E20C  23          INC   HL
E20D  B7          OR    A             <> 0 ?
E20E  C0          RET   NZ            dann unbekannt, zurück

********************************** Namen übertragen
E20F  7E          LD    A,(HL)        Byte aus Namen
E210  E6 7F       AND   7F            Endkennz. löschen
E212  CD FE E1    CALL  E1FE          in Buffer
E215  BE          CP    (HL)          war Endkennz. gesetzt ?
E216  23          INC   HL
E217  30 F6       JR    NC,E20F       nein ? dann weiter übertragen
E219  C9          RET
```

```
************************************  ggf. Space ausgeben
E21A  1D          DEC    E               Flag für Space
E21B  C0          RET    NZ              nicht gesetzt ?
E21C  3E 20       LD     A,20            sonst Space
E21E  18 DE       JR     E1FE            in Buffer

************************************  Keyword-Token nach ASCII wandeln
E220  23          INC    HL              Token übergehen
E221  FE FF       CP     FF              Funktions-Token ?
E223  20 02       JR     NZ,E227         nein ?
E225  7E          LD     A,(HL)          sonst Token laden
E226  23          INC    HL              Zeiger nach Token
E227  F5          PUSH   AF              Token retten
E228  E5          PUSH   HL              Eingabezeiger
E229  CD ED E2    CALL   E2ED            Token suchen
E22C  B7          OR     A                 Token für
E22D  28 08       JR     Z,E237            Keyword ohne Buchstabe ?
E22F  F5          PUSH   AF              1. Zeichen des Keywords
E230  CD 1A E2    CALL   E21A            ggf. Space in Buffer
E233  F1          POP    AF              1. Zeichen des Keywords
E234  CD FE E1    CALL   E1FE            in Buffer
E237  7E          LD     A,(HL)          Zeichen aus Keyword
E238  E6 7F       AND    7F              Endkennz. löschen
E23A  FE 09       CP     09              TAB ?
E23C  C4 FE E1    CALL   NZ,E1FE         nein ? dann in Buffer
E23F  BE          CP     (HL)            war Endkennz. gesetzt ?
E240  23          INC    HL
E241  28 F4       JR     Z,E237          dann weiter übertragen
E243  CD 7B FF    CALL   FF7B            letztes Keyword-Zeichen testen
E246  1E 00       LD     E,00            Flag für kein Space
E248  30 02       JR     NC,E24C         keine Ziffer, Buchstabe, "." ?
E24A  1E 01       LD     E,01            sonst Flag für folgendes Space
E24C  E1          POP    HL              Eingabezeiger
E24D  F1          POP    AF              und Token zurück
E24E  D6 E4       SUB    E4              Token für FN ?
E250  C0          RET    NZ              nein ?
E251  5F          LD     E,A             sonst Flag für kein Space
E252  C9          RET

************************************  Konstante nach ASCII wandeln
E253  D5          PUSH   DE              Bufferlänge/Space-Flag retten
E254  7E          LD     A,(HL)          Konstanten-Token
E255  23          INC    HL              Zeiger danach
E256  FE 1B       CP     1B
E258  28 49       JR     Z,E2A3          2-Byte-Binär-Konstante ?
E25A  FE 1C       CP     1C
E25C  28 50       JR     Z,E2AE          2-Byte-Hex-Konstante ?
E25E  FE 1E       CP     1E
E260  28 26       JR     Z,E288          Zeilennummer ?
E262  FE 1D       CP     1D
E264  28 22       JR     Z,E288          Zeilenadresse ?
E266  FE 1F       CP     1F
E268  28 5E       JR     Z,E2C8          REAL-Konstante ?
E26A  FE 19       CP     19
E26C  28 09       JR     Z,E277          Ein-Byte-Konstante ?
E26E  FE 1A       CP     1A
E270  28 0B       JR     Z,E27D          dez. 2-Byte-Konstante ?
E272  D6 0E       SUB    0E              Token für Konstante 0 abziehen
```

```
E274  5F        LD    E,A         gibt Konstantenwert
E275  18 02     JR    E279
E277  5E        LD    E,(HL)      Ein-Byte-Konstante laden
E278  23        INC   HL
E279  16 00     LD    D,00        Hi-Byte =0
E27B  18 04     JR    E281
E27D  5E        LD    E,(HL)
E27E  23        INC   HL          2-Byte-Konstante laden
E27F  56        LD    D,(HL)
E280  23        INC   HL
E281  E3        EX    (SP),HL     Eing.-Zg. r., Länge/Flag zur.
E282  EB        EX    DE,HL       nach DE, Konstante nach HL
E283  CD 0D FF  CALL  FF0D        Konstante in FAC eintragen
E286  18 47     JR    E2CF        und nach ASCII
E288  5E        LD    E,(HL)
E289  23        INC   HL            Zeilennummer/-adresse laden
E28A  56        LD    D,(HL)
E28B  23        INC   HL
E28C  FE 1E     CP    1E          Token für Zeilennummer ?
E28E  28 09     JR    Z,E299      dann in FAC
E290  E5        PUSH  HL          Eingabezeiger retten
E291  EB        EX    DE,HL       Zeilenadresse nach HL
E292  23        INC   HL            Zeilenende
E293  23        INC   HL            und Zeilenlänge
E294  23        INC   HL            übergehen
E295  5E        LD    E,(HL)
E296  23        INC   HL            Zeilennummer aus Zeile
E297  56        LD    D,(HL)        laden
E298  E1        POP   HL          Eingabezeiger zurück
E299  E3        EX    (SP),HL     Eing.-Zg. rett., Länge/Flag z.
E29A  EB        EX    DE,HL       nach DE, Zeilennr. nach HL
E29B  CD 0D FF  CALL  FF0D        in FAC eintragen
E29E  CD 82 EE  CALL  EE82        nach ASCII wandeln
E2A1  18 2F     JR    E2D2        und in Buffer übertragen
E2A3  C5        PUSH  BC
E2A4  01 02 00  LD    BC,0002     min. Stellenz.=0, Typ=Integer
E2A7  CD 14 F1  CALL  F114        Zahl in Binärstring wandeln
E2AA  3E 58     LD    A,58        "X" als Kennz. für Binärzahl
E2AC  18 09     JR    E2B7
E2AE  C5        PUSH  BC
E2AF  01 02 00  LD    BC,0002     min. Stellenz.=0, Typ=Integer
E2B2  CD 19 F1  CALL  F119        Zahl in Hex-String wandeln
E2B5  3E 48     LD    A,48        "H" als Kennz. für Hex-Zahl
E2B7  C1        POP   BC
E2B8  E3        EX    (SP),HL     Eing.-Zg. rett., Länge/Flag z.
E2B9  EB        EX    DE,HL       nach DE, Stringzeiger nach HL
E2BA  F5        PUSH  AF          "H"/"X"-Kennzeichen retten
E2BB  3E 26     LD    A,26        "&"
E2BD  CD FE E1  CALL  E1FE        in Buffer
E2C0  F1        POP   AF          "H"/"X"-Kennzeichen
E2C1  FE 48     CP    48          "H" ?
E2C3  C4 FE E1  CALL  NZ,E1FE     nein ? dann in Buffer
E2C6  18 0A     JR    E2D2
E2C8  3E 05     LD    A,05        Typ für REAL
E2CA  CD 4B FF  CALL  FF4B        REAL-Zahl in FAC kopieren
E2CD  E3        EX    (SP),HL     Eing.-Zg. rett., Länge/Flag z.
E2CE  EB        EX    DE,HL       nach DE
E2CF  CD 8F EE  CALL  EE8F        Integer bzw. REAL nach ASCII
```

```
E2D2  7E          LD    A,(HL)      Zeichen aus String
E2D3  23          INC   HL
E2D4  CD FE E1    CALL  E1FE        in Buffer schreiben
E2D7  7E          LD    A,(HL)      Zeichen
E2D8  B7          OR    A           Stringende ?
E2D9  20 F7       JR    NZ,E2D2     nein ? dann weiter kopieren
E2DB  E1          POP   HL          Eingabezeiger zurück
E2DC  C9          RET

*********************************  Zeiger in Keyword-Tabelle holen
                                   IN : A: Keyword-Anfangsbuchstabe
                                   OUT: DE: Zeiger auf entspr. Tab.
E2DD  E5          PUSH  HL
E2DE  D6 41       SUB   41          -"A", Nr. des Buchstaben
E2E0  87          ADD   A           mal 2, da 2 Bytes pro Eintrag
E2E1  C6 54       ADD   54
E2E3  6F          LD    L,A           $E354, Adresse der Tabelle,
E2E4  CE E3       ADC   E3            addieren
E2E6  95          SUB   L
E2E7  67          LD    H,A
E2E8  5E          LD    E,(HL)
E2E9  23          INC   HL            Tabellenzeiger aus Tabelle
E2EA  56          LD    D,(HL)        laden, nach DE
E2EB  E1          POP   HL
E2EC  C9          RET

*********************************  Token suchen, Keywordadr. holen
                                   IN : A: Token
                                   OUT: HL: Adr. des Keywords
                                        A: 1. Keyword-Buchstabe
                                        A=0, Z=0, wenn Keyword o. Buchst.
E2ED  C5          PUSH  BC
E2EE  4F          LD    C,A         Token
E2EF  06 1A       LD    B,1A        Zahl der Buchstaben
E2F1  21 88 E3    LD    HL,E388     Zeiger auf Keyword-Tabellen
E2F4  CD 13 E3    CALL  E313        Token suchen
E2F7  38 0D       JR    C,E306      gefunden ?
E2F9  23          INC   HL          Tabellenzeiger
E2FA  10 F8       DJNZ  E2F4        weitere Buchstaben ?
E2FC  21 4B E6    LD    HL,E64B     Tab. d. Keywords o. Buchstaben
E2FF  CD 13 E3    CALL  E313        Token suchen
E302  30 07       JR    NC,E30B     nicht gefunden ? dann Fehler
E304  06 C0       LD    B,C0        Flag für Keyword ohne Buchst.
E306  78          LD    A,B         1. Buchstabe des Keywords
E307  C6 40       ADD   40          ASCII-Code herstellen
E309  C1          POP   BC
E30A  C9          RET

E30B  CD 19 AC    CALL  AC19        User-Vektor
E30E  1E 02       LD    E,02        Nr. für "Syntax error"
E310  C3 94 CA    JP    CA94        Fehler ausgeben

*********************************  Token in Tabelle suchen
                                   IN : HL: Tabellenzeiger
                                        C: Token
                                   OUT: CY=1, wenn gefunden
                                        HL: Zeiger auf Keyword
E313  7E          LD    A,(HL)      Zeichen aus Tabelle
```

```
E314  B7          OR    A          Eint. dieses Buchst. zu Ende ?
E315  C8          RET   Z          dann nicht gefunden
E316  E5          PUSH  HL         Zeiger auf Keyword
E317  7E          LD    A,(HL)     nächstes Zeichen aus Keyword
E318  23          INC   HL
E319  17          RLA              Ende des Keywords erreicht ?
E31A  30 FB       JR    NC,E317    nein ? dann weiter
E31C  7E          LD    A,(HL)     zugehöriges Token
E31D  23          INC   HL         Tabellenzeiger
E31E  B9          CP    C          = gesuchtes Token ?
E31F  28 03       JR    Z,E324     dann fertig
E321  F1          POP   AF         Zeiger auf Keyword löschen
E322  18 EF       JR    E313       weitersuchen
E324  E1          POP   HL         Zeiger auf Keyword
E325  37          SCF              CY=1 für gefunden
E326  C9          RET

**********************************  String in Keyword-Tabelle suchen
                                    IN : DE: Zeiger auf Tabelle
                                         HL: Eingabezeiger
                                    OUT: DE: Zeiger auf Token
                                         CY=1, wenn gefunden
                                         C: leztes Keyword-Zeichen
                                         HL: Eingabezg. nach Keyword
                                         CY=0, wenn nicht gefunden
                                         HL: Eingabezeiger wie IN
E327  1A          LD    A,(DE)     Zeichen aus Tabelle
E328  B7          OR    A          Ende der Tabelle ?
E329  C8          RET   Z          dann nicht gefunden, zurück
E32A  E5          PUSH  HL         Eingabezeiger retten
E32B  1A          LD    A,(DE)     Zeichen aus Tabelle
E32C  13          INC   DE         Tabellenzeiger
E32D  FE 09       CP    09         TAB ?
E32F  28 04       JR    Z,E335     dann Spaces, TABs, LFs überl.
E331  FE 20       CP    20         Space ?
E333  20 05       JR    NZ,E33A    nein ?
E335  CD 61 DD    CALL  DD61       Spaces, TABs und LFs überlesen
E338  18 F1       JR    E32B       nächstes Zeichen aus Tabelle
E33A  4F          LD    C,A        Zeichen aus Keyword nach C
E33B  7E          LD    A,(HL)     Zeichen aus Eingabe
E33C  23          INC   HL
E33D  CD 8A FF    CALL  FF8A       auf Großschrift forcieren
E340  A9          XOR   C          =Zeichen aus Keyword ?
E341  28 E8       JR    Z,E32B     dann weiter vergleichen
E343  E6 7F       AND   7F         Endkennz. löschen
E345  28 0A       JR    Z,E351     letztes Zeichen ? dann gefunden
E347  1B          DEC   DE         Tabellenzeiger auf letztes Zeichen
E348  1A          LD    A,(DE)     Zeichen aus Keyword
E349  13          INC   DE
E34A  17          RLA              Ende des Keywords ?
E34B  30 FB       JR    NC,E348    nein ? dann weiter
E34D  13          INC   DE
E34E  E1          POP   HL         Zeiger auf Eingabe
E34F  18 D6       JR    E327       weitersuchen
E351  F1          POP   AF         Zeiger auf Eingabe löschen
E352  37          SCF              CY=1 für gefunden
E353  C9          RET
```

```
**********************************      Adressen der Keyword-Tabellen
E354  35 E6 2A E6 EF E5 B9 E5             Anfangsbuchstabe "A"
E35C  8A E5 7E E5 72 E5 68 E5
E364  47 E5 43 E5 3F E5 13 E5
E36C  ED E4 E2 E4 AA E4 86 E4                   bis
E374  85 E4 3B E4 FB E3 CF E3
E37C  C0 E3 B8 E3 9A E3 92 E3
E384  8D E3 88 E3                         Anfangsbuchstabe "Z"

**********************************      Basic-Keyword-Tabellen
                                          Keyword              Token Adresse
E388  4F 4E C5 DA 00                      ZONE ............. DA    F1F6
E38D  50 4F D3 48 00                      YPOS ............. 48    D10E
E392  50 4F D3 47                         XPOS ............. 47    D107
E396  4F D2 FD 00                         XOR .............. FD    FD6D
E39A  52 49 54 C5 D9                      WRITE ............ D9    F47B
E39F  49 4E 44 4F D7 D8                   WINDOW ........... D8    C2E1
E3A5  49 44 54 C8 D7                      WIDTH ............ D7    C3E3
E3AA  48 49 4C C5 D6                      WHILE ............ D6    C747
E3AF  45 4E C4 D5                         WEND ............. D5    C776
E3B3  41 49 D4 D4 00                      WAIT ............. D4    F17D
E3B8  50 4F D3 7F                         VPOS ............. 7F    C262
E3BC  41 CC 1D 00                         VAL .............. 1D    FA77
E3C0  53 49 4E C7 ED                      USING ............ ED
E3C5  50 50 45 52 A4 1C                   UPPER$ ........... 1C    F842
E3CB  4E D4 1B 00                         UNT .............. 1B    FEC2
E3CF  52 4F CE D3                         TRON ............. D3    DDE2
E3D3  52 4F 46 C6 D2                      TROFF ............ D2    DDE6
E3D8  CF EC                               TO ............... EC
E3DA  49 4D C5 46                         TIME ............. 46    D0E5
E3DE  48 45 CE EB                         THEN ............. EB
E3E2  45 53 54 D2 7D                      TESTR ............ 7D    C4EE
E3E7  45 53 D4 7C                         TEST ............. 7C    C4E9
E3EB  41 CE 1A                            TAN .............. 1A    D539
E3EE  41 47 4F 46 C6 D1                   TAGOFF ........... D1    C320
E3F4  41 C7 D0                            TAG .............. D0    C319
E3F7  41 C2 EA 00                         TAB .............. EA
E3FB  59 4D 42 4F CC CF                   SYMBOL ........... CF    F69D
E401  57 41 D0 E7                         SWAP ............. E7
E405  54 52 49 4E 47 A4 7B                STRING$ .......... 7B    FA36
E40C  54 52 A4 19                         STR$ ............. 19    F91E
E410  54 4F D0 CE                         STOP ............. CE    CB5A
E414  54 45 D0 E6                         STEP ............. E6
E418  51 D2 18                            SQR .............. 18    D4EF
E41B  D1 17                               SQ ............... 17    D329
E41D  50 45 45 C4 CD                      SPEED ............ CD    D494
E422  50 C3 E5                            SPC .............. E5
E425  50 41 43 45 A4 16                   SPACE$ ........... 16    FA57
E42B  4F 55 4E C4 CC                      SOUND ............ CC    D2C0
E430  49 CE 15                            SIN .............. 15    D52F
E433  47 CE 14                            SGN .............. 14    FF02
E436  41 56 C5 CB 00                      SAVE ............. CB    EC09
E43B  55 CE CA                            RUN .............. CA    E9BD
E43E  4F 55 4E C4 7A                      ROUND ............ 7A    D219
E443  4E C4 45                            RND .............. 45    D584
E446  49 47 48 54 A4 79                   RIGHT$ ........... 79    F943
E44C  45 54 55 52 CE C9                   RETURN ........... C9    C70F
E452  45 53 55 4D C5 C8                   RESUME ........... C8    CC03
```

```
E458  45 53 54 4F 52 C5 C7          RESTORE .......... C7   DCD9
E45F  45 4E 55 CD C6                RENUM ............ C6   E7DF
E464  45 4D 41 49 CE 13             REMAIN ........... 13   C99F
E46A  45 CD C5                      REM .............. C5   E8F3
E46D  45 4C 45 41 53 C5 C4          RELEASE .......... C4   D31E
E474  45 41 C4 C3                   READ ............. C3   DCEB
E478  41 4E 44 4F 4D 49 5A C5 C2    RANDOMIZE ........ C2   D559
E481  41 C4 C1 00                   RAD .............. C1   D4EB
E485  00
E486  52 49 4E D4 BF                PRINT ............ BF   F1FD
E48B  4F D3 78                      POS .............. 78   C276
E48E  4F 4B C5 BE                   POKE ............. BE   F15F
E492  4C 4F 54 D2 BD                PLOTR ............ BD   C4D5
E497  4C 4F D4 BC                   PLOT ............. BC   C4D0
E49B  C9 44                         PI ............... 44   D4DB
E49D  45 CE BB                      PEN .............. BB   C212
E4A0  45 45 CB 12                   PEEK ............. 12   F158
E4A4  41 50 45 D2 BA 00             PAPER ............ BA   C20A
E4AA  55 D4 B9                      OUT .............. B9   F177
E4AD  52 49 47 49 CE B8             ORIGIN ........... B8   C48C
E4B3  D2 FC                         OR ............... FC   FD63
E4B5  50 45 4E 4F 55 D4 B7          OPENOUT .......... B7   D256
E4BC  50 45 4E 49 CE B6             OPENIN ........... B6   D25F
E4C2  4E 20 53 D1 B5                ON SQ ............ B5   C940
E4C7  4E 20 45 52 52 4F 52 20 47    ON ERROR GOTO 0 .. B4   CBF8
E4D0  4F 09 54 4F 20 B0 B4
E4D7  4E 20 42 52 45 41 CB B3       ON BREAK ......... B3   C8CB
E4DF  CE B2 00                      ON ............... B2   C7E3
E4E2  4F D4 FE                      NOT .............. FE   FD77
E4E5  45 D7 B1                      NEW .............. B1   C12B
E4E8  45 58 D4 B0 00                NEXT ............. B0   C5FB
E4ED  4F 56 45 D2 AF                MOVER ............ AF   C50A
E4F2  4F 56 C5 AE                   MOVE ............. AE   C505
E4F6  4F 44 C5 AD                   MODE ............. AD   C24F
E4FA  4F C4 FB                      MOD .............. FB   FD49
E4FD  49 CE 77                      MIN .............. 77   D1EA
E500  49 44 A4 AC                   MID$ ............. AC   F993/43
E504  45 52 47 C5 AB                MERGE ............ AB   EAA6
E509  45 4D 4F 52 D9 AA             MEMORY ........... AA   F4EF
E50F  41 D8 76 00                   MAX .............. 76   D1EE
E513  4F 57 45 52 A4 11             LOWER$ ........... 11   F834
E519  4F 47 31 B0 10                LOG10 ............ 10   D525
E51E  4F C7 0F                      LOG .............. 0F   D52A
E521  4F 43 41 54 C5 A9             LOCATE ........... A9   C2D2
E527  4F 41 C4 A8                   LOAD ............. A8   E9F6
E52B  49 53 D4 A7                   LIST ............. A7   E0F7
E52F  49 4E C5 A6                   LINE ............. A6   DAF8
E533  45 D4 A5                      LET .............. A5   D654
E536  45 CE 0E                      LEN .............. 0E   FA0A
E539  45 46 54 A4 75 00             LEFT$ ............ 75   F93C
E53F  45 D9 A4 00                   KEY .............. A4   D439
E543  4F D9 0D 00                   JOY .............. 0D   D423
E547  4E D4 0C                      INT .............. 0C   FDED
E54A  4E 53 54 D2 74                INSTR ............ 74   FAA1
E54F  4E 50 55 D4 A3                INPUT ............ A3   DB2B
E554  4E D0 0B                      INP .............. 0B   F16D
E557  4E 4B 45 59 A4 43             INKEY$ ........... 43   FA24
E55D  4E 4B 45 D9 0A                INKEY ............ 0A   D409
```

```
E562  4E CB A2
E565  C6 A1 00
E568  49 4D 45 CD 42
E56D  45 58 A4 73 00
E572  4F 09 54 CF A0
E577  4F 09 53 55 C2 9F 00
E57E  52 C5 09
E581  4F D2 9E
E584  CE E4
E586  49 D8 08 00
E58A  58 D0 07
E58D  56 45 52 D9 9D
E592  52 52 4F D2 9C
E597  52 D2 41
E59A  52 CC E3
E59D  52 41 53 C5 9B
E5A2  4F C6 40
E5A5  4E D6 9A
E5A8  4E D4 99
E5AB  4E C4 98
E5AE  4C 53 C5 97
E5B2  C9 DC
E5B4  44 49 D4 96 00
E5B9  52 41 57 D2 95
E5BE  52 41 D7 94
E5C2  49 CD 93
E5C5  C9 DB
E5C7  45 4C 45 54 C5 92
E5CD  45 C7 91
E5D0  45 46 53 54 D2 90
E5D6  45 46 52 45 41 CC 8F
E5DD  45 46 49 4E D4 8E
E5E3  45 C6 8D
E5E6  45 43 A4 72
E5EA  41 54 C1 8C 00
E5EF  52 45 41 CC 06
E5F4  4F D3 05
E5F7  4F 4E D4 8B
E5FB  4C D3 8A
E5FE  4C 4F 53 45 4F 55 D4 89
E606  4C 4F 53 45 49 CE 88
E60D  4C C7 87
E610  4C 45 41 D2 86
E615  49 4E D4 04
E619  48 52 A4 03
E61D  48 41 49 CE 85
E622  41 D4 84
E625  41 4C CC 83 00
E62A  4F 52 44 45 D2 82
E630  49 4E A4 71 00
E635  55 54 CF 81
E639  54 CE 02
E63C  53 C3 01
E63F  4E C4 FA
E642  46 54 45 D2 80
E647  42 D3 00 00
```

```
INK .............. A2   C22A
IF ............... A1   C6C7
HIMEM ............ 42   D0F4
HEX$ ............. 73   F8C4
GOTO ............. A0   C6E8
GOSUB ............ 9F   C6ED
FRE .............. 09   FC2D
FOR .............. 9E   C529
FN ............... E4   D130
FIX .............. 08   FDE8
EXP .............. 07   D520
EVERY ............ 9D   C979
ERROR ............ 9C   CA8F
ERR .............. 41   D0DC
ERL .............. E3   D0EE
ERASE ............ 9B   D9C0
EOF .............. 40   C417
ENV .............. 9A   D34E
ENT .............. 99   D385
END .............. 98   CB65
ELSE ............. 97   E8F3
EI ............... DC   C8E7
EDIT ............. 96   C052
DRAWR ............ 95   C4CB
DRAW ............. 94   C4C6
DIM .............. 93   D67D
DI ............... DB   C8E1
DELETE ........... 92   E728
DEG .............. 91   D4E7
DEFSTR ........... 90   D614
DEFREAL .......... 8F   D61C
DEFINT ........... 8E   D618
DEF .............. 8D   D117
DEC$ ............. 72   F8EA
DATA ............. 8C   E8EF
CREAL ............ 06   FEEC
COS .............. 05   D534
CONT ............. 8B   CBC0
CLS .............. 8A   C25A
CLOSEOUT ......... 89   D2A1
CLOSEIN .......... 88   D298
CLG .............. 87   C4B5
CLEAR ............ 86   C132
CINT ............. 04   FE8D
CHR$ ............. 03   FA16
CHAIN ............ 85   EA3C
CAT .............. 84   D246
CALL ............. 83   F1BA
BORDER ........... 82   C221
BIN$ ............. 71   F8BA
AUTO ............. 81   C0DF
ATN .............. 02   D53E
ASC .............. 01   FA10
AND .............. FA   FD58
AFTER ............ 80   C971
ABS .............. 00   FD85
```

```
*********************************** Tabelle der Keywords ohne Buchst.
                                      Keyword           Token Adresse
E64B  DE F8                           ^ ............... F8    D4F4
E64D  DC F9                           \ ............... F9    FD37
E64F  3E 09 BD F0                     >= .............. F0
E653  3D 20 BE F0                     = > ............. F0
E657  BE EE                           > ............... EE
E659  BD EF                           = ............... EF
E65B  3C 09 BE F2                     <> .............. F2
E65F  3C 09 BD F3                     <= .............. F3
E663  3D 20 BC F3                     = < ............. F3
E667  BC F1                           < ............... F1
E669  AF F7                           / ............... F7    FD12
E66B  BA 01                           : ............... 01
E66D  AA F6                           * ............... F6    FCF5
E66F  AD F5                           - ............... F5    FCE1
E671  AB F4                           + ............... F4    FCCC
E673  A7 C0 00                        ' ............... C0    E8F3

*********************************** Programm löschen
E676  AF          XOR   A             Flag f. Zeilenadr. im Programm
E677  32 3A AE    LD    (AE3A),A      löschen
E67A  2A 81 AE    LD    HL,(AE81)     Zeiger auf Programmstart
E67D  77          LD    (HL),A          Null als 1. Zeilenende
E67E  23          INC   HL
E67F  77          LD    (HL),A          2 Nullen als
E680  23          INC   HL              Kennzeichen für Programmende
E681  77          LD    (HL),A
E682  23          INC   HL
E683  22 83 AE    LD    (AE83),HL     Zeiger auf Programmende setzen
E686  C9          RET

*********************************** Zeilenadressen eliminieren
E687  3A 3A AE    LD    A,(AE3A)        keine Zeilenadressen
E68A  B7          OR    A               im Programm ?
E68B  C8          RET   Z             dann fertig
E68C  C5          PUSH  BC
E68D  D5          PUSH  DE
E68E  E5          PUSH  HL
E68F  01 9D E6    LD    BC,E69D       Adr. f. Zeilenadr. wandeln
E692  CD FF E8    CALL  E8FF          Programm durchg., Rout. ausf.
E695  AF          XOR   A               Flag für keine Zeilenadr.
E696  32 3A AE    LD    (AE3A),A        im Programm setzen
E699  E1          POP   HL
E69A  D1          POP   DE
E69B  C1          POP   BC
E69C  C9          RET

*********************************** Zeilenadr. durch Zeilennr. ers.
E69D  CD 43 E9    CALL  E943          nächstes Item
E6A0  FE 02       CP    02            Statementende ?
E6A2  D8          RET   C             dann zurück
E6A3  FE 1D       CP    1D            Token für Zeilenadresse
E6A5  20 F6       JR    NZ,E69D       nein ? dann weiter suchen
E6A7  56          LD    D,(HL)
E6A8  2B          DEC   HL              Zeilenadresse laden
E6A9  5E          LD    E,(HL)
E6AA  2B          DEC   HL
```

```
E6AB  E5          PUSH  HL          Zeiger auf Token
E6AC  EB          EX    DE,HL       Zeilenadresse nach HL
E6AD  23          INC   HL            Zeilenende
E6AE  23          INC   HL            und Zeilenlänge
E6AF  23          INC   HL            übergehen
E6B0  5E          LD    E,(HL)
E6B1  23          INC   HL          Zeilennummer
E6B2  56          LD    D,(HL)      laden
E6B3  E1          POP   HL          Zeiger auf Token
E6B4  36 1E       LD    (HL),1E     Token für Zeilennr. setzen
E6B6  23          INC   HL
E6B7  73          LD    (HL),E        Zeilennummer
E6B8  23          INC   HL            ins Programm eintragen
E6B9  72          LD    (HL),D
E6BA  18 E1       JR    E69D        weiter suchen

*********************************** Eingabezeile auswerten
                                  OUT: CY=0, Z=0 für Znr.-Überlauf
                                         CY=0, Z=1 für Direkteingabe
                                         CY=1, Z=0 f. Zeile eingefügt
                                         CY=1, Z=1 f. sofortiges Ende
E6BC  CD 61 DD    CALL  DD61        Spaces, TABs und LFs überlesen
E6BF  B7          OR    A           Zeilenende ?
E6C0  37          SCF               CY=1 für sofortiges Ende
E6C1  C8          RET   Z           Zeilenende ? dann zurück
E6C2  CD 04 EE    CALL  EE04        Zeilennr.-String wandeln
E6C5  D0          RET   NC          keine Zeilennummer ?
E6C6  7E          LD    A,(HL)      nächstes Zeichen
E6C7  FE 20       CP    20          Space ?
E6C9  20 01       JR    NZ,E6CC     nein ?
E6CB  23          INC   HL          sonst Space übergehen
E6CC  CD D2 E6    CALL  E6D2        Zeile ins Programm einfügen
E6CF  37          SCF               CY=1,
E6D0  9F          SBC   A           Z=0 für Zeile eingefügt
E6D1  C9          RET

*********************************** Zeile im Programm einfügen
                                  IN : DE: Zeilennr.
                                         HL: Zeiger auf Zeilentext
E6D2  CD 87 E6    CALL  E687          Zeilenadressen eliminieren
E6D5  CD BB DE    CALL  DEBB        Zeile tokenisieren
E6D8  E5          PUSH  HL          Zeiger auf tokenisierte Zeile
E6D9  CD 61 DD    CALL  DD61        Spaces, TABs und LFs überlesen
E6DC  B7          OR    A           Zeilenende ?
E6DD  28 28       JR    Z,E707      dann nur alte Zeile löschen
E6DF  C5          PUSH  BC          Zeilentextlänge
E6E0  D5          PUSH  DE          Zeilennummer
E6E1  21 04 00    LD    HL,0004     4 Bytes f. Zeilenlänge/-nummer
E6E4  09          ADD   HL,BC       addieren
E6E5  E5          PUSH  HL            Gesamtzeilenlänge
E6E6  E5          PUSH  HL            retten
E6E7  CD A3 E7    CALL  E7A3        Zeile im Programm suchen
E6EA  E5          PUSH  HL          Adresse der (nächsten) Zeile
E6EB  DC 0B E7    CALL  C,E70B      Zeile gef. ? dann löschen
E6EE  D1          POP   DE          Einfügeadresse
E6EF  C1          POP   BC          Gesamtlänge der neuen Zeile
E6F0  CD F8 F5    CALL  F5F8        Platz für Zeile schaffen
E6F3  CD 2C F5    CALL  F52C        Programm/Var.-Zeiger korrig.
```

```
E6F6  EB          EX    DE,HL       Einfügeadresse nach HL
E6F7  D1          POP   DE          Gesamtzeilenlänge
E6F8  73          LD    (HL),E
E6F9  23          INC   HL            in Zeile eintragen
E6FA  72          LD    (HL),D
E6FB  23          INC   HL
E6FC  D1          POP   DE          Zeilennummer
E6FD  73          LD    (HL),E
E6FE  23          INC   HL            in Zeile eintragen
E6FF  72          LD    (HL),D
E700  23          INC   HL
E701  C1          POP   BC          Zeilentextlänge
E702  EB          EX    DE,HL       Adr. für Text in neuer Zeile
E703  E1          POP   HL          Zeiger auf Zeilentext
E704  C3 F2 FF    JP    FFF2        tokenisierten Text kopieren

E707  E1          POP   HL          Zeiger auf neue Zeile löschen
E708  CD 9A E7    CALL  E79A        alte Zeile im Programm suchen

*********************************** Bereich aus Programm löschen
                                    IN : HL: Adresse des Bereichs
                                         BC: Länge des Bereichs
E70B  C5          PUSH  BC            Länge
E70C  E5          PUSH  HL            und Adresse d. Bereichs retten
E70D  09          ADD   HL,BC         Länge addieren, gibt Endadr.
E70E  EB          EX    DE,HL         Endadresse (+1) nach DE
E70F  2A 89 AE    LD    HL,(AE89)     Zeiger auf Ende der Felder
E712  CD CF FF    CALL  FFCF          minus Endadresse des Bereichs
E715  44          LD    B,H             gibt zu verschiebende
E716  4D          LD    C,L             Länge, nach BC
E717  EB          EX    DE,HL         Endadresse des Bereichs n. HL
E718  D1          POP   DE            Startadresse nach DE
E719  78          LD    A,B
E71A  B1          OR    C               Länge <>0 ?
E71B  C4 F2 FF    CALL  NZ,FFF2       dann Programm/Var. verschieben
E71E  D1          POP   DE            Länge des gelöschten Bereichs
E71F  21 00 00    LD    HL,0000       Null
E722  CD DA FF    CALL  FFDA          -Länge gibt Offset
E725  C3 2C F5    JP    F52C          Off. zu Prg./Var.-Zeigern add.

*********************************** Basic-Befehl DELETE
E728  CD 37 E7    CALL  E737          zu löschenden Bereich holen
E72B  CD 4A DD    CALL  DD4A          auf Statementende prüfen
E72E  CD 5A E7    CALL  E75A          Programmbereich löschen
E731  CD 7A C1    CALL  C17A          Basic-Zeiger initialisieren
E734  C3 64 C0    JP    C064          zur Eingabeschleife

*********************************** Löschbereich für DELETE holen
E737  CD B0 CE    CALL  CEB0          Zeilennummernbereich holen
E73A  E5          PUSH  HL            Basic-PC
E73B  C5          PUSH  BC            und Startzeilennr. retten
E73C  CD C1 E7    CALL  E7C1          Zeile nach Endzeilennr. suchen
E73F  D1          POP   DE            Startzeilennr.
E740  E5          PUSH  HL            Zeiger nach Endzeile
E741  CD A3 E7    CALL  E7A3          Startzeile im Programm suchen
E744  22 3B AE    LD    (AE3B),HL     Zeiger a. Startzeile=Startadr.
E747  EB          EX    DE,HL         Startadresse nach DE
E748  E1          POP   HL            Zeiger nach Endzeile
```

```
E749  CD CF FF     CALL   FFCF          Startadresse abziehen
E74C  22 3D AE     LD     (AE3D),HL     gibt zu löschende Länge
E74F  38 04        JR     C,E755        Startadr.> Endadr. ? d. Fehler
E751  7C           LD     A,H             zu löschende Länge
E752  B5           OR     L               ungleich Null ?
E753  E1           POP    HL            Basic-PC zurück
E754  C0           RET    NZ            Länge <>0 ? dann o.k.
E755  1E 05        LD     E,05          Nr. für "Improper argument"
E757  C3 94 CA     JP     CA94          Fehler ausgeben

*********************************** Programmbereich f. DELETE löschen
E75A  CD 87 E6     CALL   E687            Zeilenadressen eliminieren
E75D  ED 4B 3D AE  LD     BC,(AE3D)       zu löschende Länge
E761  2A 3B AE     LD     HL,(AE3B)       Start-Löschadresse
E764  C3 0B E7     JP     E70B            Programmbereich löschen

*********************************** Zeilenadresse holen
                                    IN : HL: PC auf Token
                                          A: Token
                                    OUT: DE: Zeilenadresse
                                          beim CPC 664/6128:
                                          A: folgendes Zeichen
                                          Z=1, wenn Statementende
E767  23           INC    HL            Zeiger nach Token
E768  5E           LD     E,(HL)
E769  23           INC    HL              Zeilennummer bzw.
E76A  56           LD     D,(HL)          Zeilenadresse laden
E76B  23           INC    HL
E76C  FE 1D        CP     1D            Token für Zeilenadresse ?
E76E  C8           RET    Z             dann fertig
E76F  FE 1E        CP     1E            Token für Zeilennummer ?
E771  C2 EA E8     JP     NZ,E8EA       nein ? dann "Syntax error"
E774  E5           PUSH   HL            Basic-PC retten
E775  CD D6 DD     CALL   DDD6          akt. Zeilennummer holen
E778  DC B8 FF     CALL   C,FFB8        mit gesuchter vergleichen
E77B  30 09        JR     NC,E786       akt. Nr. größer ? d. ab Start
E77D  E1           POP    HL              Basic-PC
E77E  E5           PUSH   HL
E77F  CD F3 E8     CALL   E8F3          Rest der akt. Zeile überlesen
E782  23           INC    HL            Null am Zeilenende übergehen
E783  CD A7 E7     CALL   E7A7          ab dort Zeile suchen
E786  D4 9A E7     CALL   NC,E79A       ggf. Zeile ab Pg.-Start suchen
E789  2B           DEC    HL            Zeiger auf Null vor Zeile
E78A  EB           EX     DE,HL         nach DE
E78B  E1           POP    HL              Basic-PC
E78C  E5           PUSH   HL
E78D  3E 1D        LD     A,1D          Token für Zeilenadresse
E78F  32 3A AE     LD     (AE3A),A      Flag für Zeilenadressen setzen
E792  2B           DEC    HL
E793  72           LD     (HL),D
E794  2B           DEC    HL              Zeilenadresse
E795  73           LD     (HL),E
E796  2B           DEC    HL
E797  77           LD     (HL),A        und Token ins Programm eintr.
E798  E1           POP    HL            PC nach Zeilenadresse
E799  C9           RET
```

```
**********************************  Zeile suchen, ggf. Fehler ausgeb.
                                    IN : DE: Zeilenadresse
                                    OUT: HL: Zeiger auf Zeile
                                         BC: Zeilenlänge
E79A  CD A3 E7   CALL   E7A3          Zeile im Programm suchen
E79D  D8         RET    C             gefunden ?
E79E  1E 08      LD     E,08          Nr. für "Line does not exist"
E7A0  C3 94 CA   JP     CA94          Fehler ausgeben

**********************************  Zeile im Programm suchen
                                    IN : DE: Zeilenadresse
                                    OUT: HL: Zeiger auf Zeile
                                             oder nächste Zeile
                                         BC: Zeilenlänge
                                         CY=1, wenn gefunden
E7A3  2A 81 AE   LD     HL,(AE81)     Zeiger auf Programmstart
E7A6  23         INC    HL            Null am Programmstart überl.
E7A7  4E         LD     C,(HL)
E7A8  23         INC    HL              nächste Zeilenlänge
E7A9  46         LD     B,(HL)          laden
E7AA  2B         DEC    HL
E7AB  78         LD     A,B             Zeilenlänge =0 ?
E7AC  B1         OR     C               (Programmende ?)
E7AD  C8         RET    Z             dann nicht gefunden (CY=0)
E7AE  E5         PUSH   HL            Zeiger auf Zeile retten
E7AF  23         INC    HL
E7B0  23         INC    HL              Zeiger auf Zeilennummer
E7B1  7E         LD     A,(HL)
E7B2  23         INC    HL              Zeilennummer laden
E7B3  66         LD     H,(HL)
E7B4  6F         LD     L,A
E7B5  EB         EX     DE,HL
E7B6  CD B8 FF   CALL   FFB8            m. gesuchter Nr. vergleichen
E7B9  EB         EX     DE,HL
E7BA  E1         POP    HL            Zeiger auf Zeile
E7BB  3F         CCF
E7BC  D0         RET    NC            akt. Nr. > gesuchte Nr. ?
E7BD  C8         RET    Z             gleich ? dann gefunden (CY=1)
E7BE  09         ADD    HL,BC         sonst Zeilenlänge addieren
E7BF  18 E6      JR     E7A7          weiter suchen

**********************************  nächsthöhere Zeile suchen
                                    IN : DE: Zeilennummer
                                         HL: Zeiger auf nächste Zeile
                                         BC: Zeilenlänge
E7C1  2A 81 AE   LD     HL,(AE81)     Zeiger auf Programmstart
E7C4  23         INC    HL            Null am Programmstart überles.
E7C5  E5         PUSH   HL            Zeiger auf Zeile retten
E7C6  4E         LD     C,(HL)
E7C7  23         INC    HL              Zeilenlänge laden
E7C8  46         LD     B,(HL)
E7C9  23         INC    HL
E7CA  78         LD     A,B             Zeilenlänge =0
E7CB  B1         OR     C               (Programmende ?)
E7CC  28 0F      JR     Z,E7DD        dann fertig
E7CE  7E         LD     A,(HL)
E7CF  23         INC    HL              sonst Zeilennr. laden
E7D0  66         LD     H,(HL)
```

```
E7D1  6F        LD    L,A
E7D2  EB        EX    DE,HL
E7D3  CD B8 FF  CALL  FFB8       mit gesuchter Nr. vergleichen
E7D6  EB        EX    DE,HL
E7D7  38 04     JR    C,E7DD     Nr. > gesuchte Nr. ? d. fertig
E7D9  E1        POP   HL         sonst Zeiger auf Zeile zurück
E7DA  09        ADD   HL,BC      Zeilenlänge addieren
E7DB  18 E8     JR    E7C5       weiter suchen
E7DD  E1        POP   HL         Zeiger auf Zeile
E7DE  C9        RET

*********************************** Basic-Befehl RENUM
E7DF  11 0A 00  LD    DE,000A      Default f. neue Startzeilennr.
E7E2  28 05     JR    Z,E7E9       Statementende ? dann Default
E7E4  FE 2C     CP    2C           "," ?
E7E6  C4 E1 CE  CALL  NZ,CEE1      nein ? dann Zeilennr. holen
E7E9  D5        PUSH  DE           neue Startzeilennummer retten
E7EA  11 00 00  LD    DE,0000      Default f. alte Startzeilennr.
E7ED  CD 55 DD  CALL  DD55         folgt Komma ?
E7F0  30 05     JR    NC,E7F7      nein ? dann Default
E7F2  FE 2C     CP    2C           zweites Komma ?
E7F4  C4 E1 CE  CALL  NZ,CEE1      nein ? dann Zeilennr. holen
E7F7  D5        PUSH  DE           alte Startzeilennummer retten
E7F8  11 0A 00  LD    DE,000A      Default für Schrittweite
E7FB  CD 55 DD  CALL  DD55         folgt Komma ?
E7FE  DC E1 CE  CALL  C,CEE1       dann Schrittweite holen
E801  CD 4A DD  CALL  DD4A         auf Statementende prüfen
E804  E1        POP   HL           alte Startzeilennr.
E805  EB        EX    DE,HL          Schrittweite retten,
E806  E3        EX    (SP),HL        neue Startzeilennummer
E807  EB        EX    DE,HL          vom Stack
E808  D5        PUSH  DE           neue Startzeilennr.
E809  E5        PUSH  HL           alte Startzeilennr.
E80A  CD A3 E7  CALL  E7A3         neue Startzeilennr. suchen
E80D  D1        POP   DE           alte Startzeilennr.
E80E  E5        PUSH  HL           neue Startzeilenadresse
E80F  CD A3 E7  CALL  E7A3         alte Startzeilennr. suchen
E812  EB        EX    DE,HL        alte Startzeilenadr. nach DE
E813  E1        POP   HL           neue Startzeilenadresse
E814  CD B8 FF  CALL  FFB8         alte Startzeilenadr. größer ?
E817  DA 55 E7  JP    C,E755       dann "Improper argument"
E81A  EB        EX    DE,HL        alte Startzeilenadr. nach HL
E81B  D1        POP   DE           neue Startzeilennr.
E81C  C1        POP   BC           Schrittweite
E81D  D5        PUSH  DE           neue Startzeilennr.,
E81E  E5        PUSH  HL           alte Startzeilenadr.,
E81F  C5        PUSH  BC           Schrittweite retten
E820  4E        LD    C,(HL)
E821  23        INC   HL             Zeilenlänge laden
E822  46        LD    B,(HL)
E823  78        LD    A,B            Zeilenlänge =0 ?
E824  B1        OR    C              (Programmende ?)
E825  28 13     JR    Z,E83A       dann mit RENUM beginnen
E827  2B        DEC   HL           Zeiger auf Zeile
E828  09        ADD   HL,BC        Länge addieren
E829  7E        LD    A,(HL)         nächste
E82A  23        INC   HL             Zeilenlänge =0 ?
E82B  B6        OR    (HL)           (Programmende ?)
```

```
E82C  28 0C      JR    Z,E83A       dann mit RENUM beginnen
E82E  2B         DEC   HL           Zeiger auf Zeile
E82F  C1         POP   BC           Schrittweite
E830  E5         PUSH  HL           Zeiger auf Zeile
E831  EB         EX    DE,HL          Differenz zu
E832  09         ADD   HL,BC          neuer Zeilennummer
E833  EB         EX    DE,HL          addieren
E834  DA 55 E7   JP    C,E755       Überlauf ? dann Fehler
E837  E1         POP   HL           Zeiger auf Zeile
E838  18 E5      JR    E81F         weiter prüfen
E83A  01 64 E8   LD    BC,E864      Adr. für Zeilennr. ersetzen
E83D  CD FF E8   CALL  E8FF         Programm durchgehen, ersetzen
E840  C1         POP   BC           Schrittweite
E841  E1         POP   HL           alte Startzeilenadresse
E842  D1         POP   DE           neue Startzeilennummer
E843  C5         PUSH  BC           Schrittweite
E844  E5         PUSH  HL           Zeiger auf Zeile
E845  4E         LD    C,(HL)
E846  23         INC   HL             Zeilenlänge laden
E847  46         LD    B,(HL)
E848  23         INC   HL
E849  78         LD    A,B            Zeilenlänge =0 ?
E84A  B1         OR    C              (Programmende ?)
E84B  28 0C      JR    Z,E859       dann fertig numeriert
E84D  73         LD    (HL),E
E84E  23         INC   HL             neue Zeilennumer in
E84F  72         LD    (HL),D         Zeile speichern
E850  23         INC   HL
E851  E1         POP   HL           Zeiger auf Zeile
E852  09         ADD   HL,BC        Länge addieren
E853  C1         POP   BC           Schrittweite
E854  EB         EX    DE,HL
E855  09         ADD   HL,BC          zu Zeilennr. addieren
E856  EB         EX    DE,HL
E857  18 EA      JR    E843         nächste Zeile
E859  E1         POP   HL           Zeiger auf Zeile
E85A  E1         POP   HL           und Schrittweite vom Stack
E85B  01 88 E8   LD    BC,E888      Adr. für Fehler bei Zeilennr.
E85E  CD FF E8   CALL  E8FF         Programm durchgehen
E861  C3 64 C0   JP    C064         zur Eingabeschleife

********************************** Zeilennr. im Statement ersetzen
E864  CD 43 E9   CALL  E943         nächstes Item holen
E867  FE 02      CP    02           Statementende ?
E869  D8         RET   C            dann zurück
E86A  FE 1E      CP    1E           Token für Zeilennummer ?
E86C  20 F6      JR    NZ,E864      nein ? dann weiter suchen
E86E  E5         PUSH  HL           Zeiger auf Zeilennr. hi
E86F  56         LD    D,(HL)
E870  2B         DEC   HL             Zeilennumer laden
E871  5E         LD    E,(HL)
E872  CD A3 E7   CALL  E7A3         Zeile im Programm suchen
E875  30 0E      JR    NC,E885      nicht gefunden ? dann weiter
E877  2B         DEC   HL           Zeiger auf Null vor Zeile
E878  EB         EX    DE,HL        nach DE
E879  E1         POP   HL           Zeiger auf Zeilennr. hi
E87A  E5         PUSH  HL
E87B  72         LD    (HL),D         Zeilennummer durch
```

```
E87C  2B        DEC   HL          Zeilenadresse
E87D  73        LD    (HL),E      (Zeiger auf Null vor Zeile)
E87E  2B        DEC   HL          ersetzen
E87F  3E 1D     LD    A,1D       Token für Zeilenadresse
E881  77        LD    (HL),A     setzen
E882  32 3A AE  LD    (AE3A),A   Flag für Zeilenadressen setzen
E885  E1        POP   HL         Suchzeiger
E886  18 DC     JR    E864       Statement weiter durchgehen

**********************************  bei Znr. im Statem. Fehler ausg.
E888  CD 43 E9  CALL  E943       nächstes Item holen
E88B  FE 02     CP    02         Statementende ?
E88D  D8        RET   C          dann zurück
E88E  FE 1E     CP    1E         Token für Zeilennummer ?
E890  20 F6     JR    NZ,E888    nein ? dann weiter suchen
E892  E5        PUSH  HL         Zeiger auf Zeilennr. hi
E893  56        LD    D,(HL)
E894  2B        DEC   HL          Zeilennummer laden
E895  5E        LD    E,(HL)
E896  CD D6 DD  CALL  DDD6       akt. Zeilennr. holen (??)
E899  CD 18 CB  CALL  CB18       "undefined line xxxx in yyyy"
E89C  E1        POP   HL         Suchzeiger
E89D  18 E9     JR    E888       Statement weiter durchgehen

**********************************  zugehöriges ELSE suchen
                                  IN : HL: PC
                                  OUT: Z=0 für ELSE gefunden
                                       HL: Zeiger nach ELSE-Token
                                       Z=1 für kein ELSE gefunden
                                       HL: Zeiger auf nächste Zeile
E89F  06 01     LD    B,01       Verschachtelungstiefe
E8A1  2B        DEC   HL         PC eins zurück
E8A2  CD 43 E9  CALL  E943       nächstes Item holen
E8A5  B7        OR    A          Zeilenende ?
E8A6  C8        RET   Z          dann nicht gefunden, Z=1
E8A7  FE 01     CP    01         Statementende (":") ?
E8A9  28 07     JR    Z,E8B2     dann auf ELSE prüfen
E8AB  FE A1     CP    A1         Token für IF ?
E8AD  20 F3     JR    NZ,E8A2    nein ? dann weiter suchen
E8AF  04        INC   B          Verschachtelungstiefe erhöhen
E8B0  18 F0     JR    E8A2       weiter suchen
E8B2  CD 43 E9  CALL  E943       nächstes Item holen
E8B5  FE 97     CP    97         Token für ELSE ?
E8B7  20 EC     JR    NZ,E8A5    nein ? dann weiter suchen
E8B9  05        DEC   B          Verschachtelungstiefe
E8BA  20 E6     JR    NZ,E8A2    weitere Verschachtelungen ?
E8BC  CD 3F DD  CALL  DD3F       Zeiger nach ELSE-Token
E8BF  04        INC   B          Z=0 für ELSE gefunden
E8C0  C9        RET

**********************************  Arrayindizes ggf. überlesen
E8C1  7E        LD    A,(HL)     akt. Zeichen
E8C2  FE 5B     CP    5B         eckige Klammer auf ?
E8C4  28 03     JR    Z,E8C9     dann Indizes überlesen
E8C6  FE 28     CP    28         runde Klammer auf ?
E8C8  C0        RET   NZ         nein ? dann kein Array
E8C9  06 00     LD    B,00       Klammer-Verschachtelungstiefe
E8CB  04        INC   B          erhöhen
```

```
E8CC  CD 43 E9    CALL  E943        nächstes Item holen
E8CF  FE 5B       CP    5B          eckige Klammer auf ?
E8D1  28 F8       JR    Z,E8CB      dann Tiefe erhöhen
E8D3  FE 28       CP    28          runde Klammer auf ?
E8D5  28 F4       JR    Z,E8CB      dann Tiefe erhöhen
E8D7  FE 5D       CP    5D          eckige Klammer zu ?
E8D9  28 0A       JR    Z,E8E5      dann Tiefe herunterzählen
E8DB  FE 29       CP    29          runde Klammer zu ?
E8DD  28 06       JR    Z,E8E5      dann Tiefe herunterzählen
E8DF  FE 02       CP    02          Statementende ?
E8E1  38 07       JR    C,E8EA      dann "Syntax error"
E8E3  18 E7       JR    E8CC        sonst weiter suchen
E8E5  05          DEC   B           Verschachtelungstiefe
E8E6  20 E4       JR    NZ,E8CC     weitere Verschachtelungen ?
E8E8  23          INC   HL          Zeiger nach Indizes
E8E9  C9          RET

E8EA  1E 02       LD    E,02        Nr. für "Syntax error"
E8EC  C3 94 CA    JP    CA94        Fehler ausgeben

*********************************** Basic-Befehl DATA
E8EF  06 01       LD    B,01          Trennzeichen, ":"
E8F1  18 02       JR    E8F5          nächstes Statement suchen

*********************************** Basic-Befehle REM, ELSE, '
E8F3  06 00       LD    B,00          Trennzeichen = Zeilenende
E8F5  2B          DEC   HL
E8F6  CD 43 E9    CALL  E943          nächstes Item holen
E8F9  B7          OR    A             Zeilenende ?
E8FA  C8          RET   Z             dann zurück
E8FB  B8          CP    B             Trennzeichen ?
E8FC  20 F8       JR    NZ,E8F6       nein ? dann weiter suchen
E8FE  C9          RET

*********************************** Programm durchgehen, Routine aus.
                                    IN : BC: Adresse der Routine
                                           (Routine für ein Statement)
E8FF  CD D2 DD    CALL  DDD2          aktuelle Zeilenadresse holen
E902  E5          PUSH  HL            und retten
E903  2A 81 AE    LD    HL,(AE81)     Zeiger auf Programmstart
E906  23          INC   HL            Null am Zeilenende übergehen
E907  7E          LD    A,(HL)
E908  23          INC   HL              nächste Zeilenlänge =0 ?
E909  B6          OR    (HL)            (Programmende ?)
E90A  28 13       JR    Z,E91F        dann fertig
E90C  23          INC   HL            Zeiger auf Zeilennummer
E90D  CD CE DD    CALL  DDCE          als akt. Zeilenadresse setzen
E910  23          INC   HL            Zeiger auf Zeilennummer hi
E911  C5          PUSH  BC            Adresse der Routine retten
E912  CD F9 FF    CALL  FFF9          Routine ausführen
E915  C1          POP   BC            Adresse der Routine zurück
E916  2B          DEC   HL            Zg. auf letztes bearb. Zeichen
E917  CD 35 E9    CALL  E935          bis Statementende/THEN/ELSE
E91A  B7          OR    A             Zeilenende ?
E91B  20 F4       JR    NZ,E911       nein ? dann nächstes Statement
E91D  18 E7       JR    E906          sonst nächste Zeile
E91F  E1          POP   HL            alte Zeilenadresse
E920  C3 CE DD    JP    DDCE          wieder als akt. Zeilenadresse
```

```
***********************************   nächstes Statement, ggf. Fehler
                                      IN : C: Nr. des Fehlers
                                      IN/OUT: DE: akt. Zeilenadresse
E923  CD 35 E9    CALL   E935            Statementende/THEN/ELSE suchen
E926  B7          OR     A               Zeilenende ?
E927  C0          RET    NZ              nein ? dann o.k.
E928  23          INC    HL
E929  7E          LD     A,(HL)
E92A  23          INC    HL                nächste Zeilenlänge =0 ?
E92B  B6          OR     (HL)              (Programmende ?)
E92C  59          LD     E,C             Nr. des Fehlers
E92D  CA 94 CA    JP     Z,CA94          Programmende ? dann Fehler
E930  23          INC    HL              Zeiger auf Zeilennummer
E931  54          LD     D,H               nach DE
E932  5D          LD     E,L
E933  23          INC    HL              Zeiger vor Zeilentext
E934  C9          RET

***********************************   Statementende/THEN/ELSE suchen
E935  CD 43 E9    CALL   E943            nächstes Item holen
E938  FE 02       CP     02              Statementende ?
E93A  D8          RET    C               dann zurück
E93B  FE 97       CP     97              Token für ELSE ?
E93D  C8          RET    Z               dann zurück
E93E  FE EB       CP     EB              Token für THEN ?
E940  20 F3       JR     NZ,E935         nein ? dann weiter suchen
E942  C9          RET

***********************************   nächstes Item suchen
                                      IN : HL: PC vor Item
                                      OUT: HL: PC vor nächstem Item
                                            A: Token dieses Items
E943  CD 3F DD    CALL   DD3F            1. Zeichen des Items holen
E946  C8          RET    Z               Statementende ? dann zurück
E947  FE 0E       CP     0E              Token für Variable ?
E949  38 1D       JR     C,E968          dann Namen überlesen
E94B  FE 20       CP     20              Token für Konstante ?
E94D  38 29       JR     C,E978          dann Wert überlesen
E94F  FE 22       CP     22              '"' ?
E951  28 09       JR     Z,E95C          dann String überlesen
E953  FE 7C       CP     7C              RSX-Token ?
E955  28 19       JR     Z,E970          dann RSX-Wort überlesen
E957  FE FF       CP     FF              Funktions-Token ?
E959  C0          RET    NZ              nein ? dann zurück
E95A  23          INC    HL              sonst Zeiger auf Token
E95B  C9          RET

***********************************   String überlesen
                                      OUT: A: $22
E95C  23          INC    HL              nächstes Zeichen
E95D  7E          LD     A,(HL)          laden
E95E  FE 22       CP     22              '"' ?
E960  C8          RET    Z               dann zurück
E961  B7          OR     A               Zeilenende ?
E962  20 F8       JR     NZ,E95C         nein ? dann weitersuchen
E964  2B          DEC    HL              Zeiger vor Zeilenende
E965  3E 22       LD     A,22            '"'
E967  C9          RET
```

```
*********************************   Variable überlesen
E968  FE 08       CP    08
E96A  C8          RET   Z            (??)
E96B  FE 07       CP    07
E96D  C8          RET   Z
E96E  23          INC   HL             Token/Offset
E96F  23          INC   HL             übergehen
E970  F5          PUSH  AF           Token retten
E971  23          INC   HL           Zeiger auf Namen
E972  7E          LD    A,(HL)       Byte aus Namen
E973  17          RLA                Name zu Ende ?
E974  30 FB       JR    NC,E971      nein ? dann weiter
E976  F1          POP   AF           Variablen-Token
E977  C9          RET

*********************************   Konstante überlesen
E978  FE 18       CP    18           Kurz-Konstante ?
E97A  D8          RET   C            dann zurück
E97B  FE 19       CP    19
E97D  28 08       JR    Z,E987       Ein-Byte-Konstante ?
E97F  FE 1F       CP    1F           REAL-Konstante ?
E981  38 03       JR    C,E986       nein ? dann Integer-Konstante
E983  23          INC   HL
E984  23          INC   HL
E985  23          INC   HL
E986  23          INC   HL
E987  23          INC   HL
E988  C9          RET

*********************************   Variablenoffsets löschen
E989  C5          PUSH  BC
E98A  D5          PUSH  DE
E98B  E5          PUSH  HL
E98C  01 96 E9    LD    BC,E996      Adr. f. Offsets löschen
E98F  CD FF E8    CALL  E8FF         Programm durchgehen
E992  E1          POP   HL
E993  D1          POP   DE
E994  C1          POP   BC
E995  C9          RET

*********************************   Offsets im Statement löschen
E996  E5          PUSH  HL           Zeiger vor Item
E997  CD 43 E9    CALL  E943         nächstes Item holen
E99A  D1          POP   DE           Zeiger vor Item davor
E99B  FE 02       CP    02           Statementende ?
E99D  D8          RET   C            dann zurück
E99E  FE 0E       CP    0E           keine Variable ?
E9A0  30 F4       JR    NC,E996      dann weiter suchen
E9A2  FE 07       CP    07
E9A4  28 F0       JR    Z,E996       (??)
E9A6  FE 08       CP    08
E9A8  28 EC       JR    Z,E996
E9AA  EB          EX    DE,HL        Zeiger vor Variable nach HL
E9AB  CD 3F DD    CALL  DD3F         Variablen-Token holen
E9AE  FE 0D       CP    0D
E9B0  38 02       JR    C,E9B4       markierte Variable ?
E9B2  36 0D       LD    (HL),0D      sonst Token f. Var. o. Kennz.
E9B4  23          INC   HL
```

```
E9B5  36 00        LD     (HL),00
E9B7  23           INC    HL          Variablenoffset löschen
E9B8  36 00        LD     (HL),00
E9BA  EB           EX     DE,HL      Suchzeiger wieder nach HL
E9BB  18 D9        JR     E996       weiter durchgehen

*********************************** Basic-Befehl RUN
E9BD  CD 51 DD     CALL   DD51       Statementende ?
E9C0  EB           EX     DE,HL
E9C1  2A 81 AE     LD     HL,(AE81)   Programmstart nach DE
E9C4  EB           EX     DE,HL
E9C5  38 1C        JR     C,E9E3     dann von Programmstart ab
E9C7  FE 1E        CP     1E         Token für Zeilennummer ?
E9C9  28 15        JR     Z,E9E0     dann ab Programmzeile
E9CB  FE 1D        CP     1D         Token für Zeilenadresse ?
E9CD  28 11        JR     Z,E9E0     dann ab Programmzeile
E9CF  CD 0D EA     CALL   EA0D       1. Block lesen und auswerten
E9D2  21 30 EA     LD     HL,EA30    Adr. f. Binärdatei laden
E9D5  D2 13 BD     JP     NC,BD13    Binärdat. ? d. MC BOOT PROGRAM
E9D8  CD A8 EB     CALL   EBA8       sonst Programm laden
E9DB  2A 81 AE     LD     HL,(AE81)  Zeiger auf Programmstart
E9DE  18 11        JR     E9F1       Programm starten

E9E0  CD 67 E7     CALL   E767       Zeilenadresse holen
E9E3  D5           PUSH   DE         Zeiger auf Einsprung retten
E9E4  CD AD D2     CALL   D2AD       Kassette initialisieren
E9E7  CD 8C C1     CALL   C18C       Variablen löschen
E9EA  CD 7A C1     CALL   C17A       Basic-Zeiger initialisieren
E9ED  CD 5E C1     CALL   C15E       Ausdruckausw. und I/O init.
E9F0  E1           POP    HL         neuer Programmzeiger
E9F1  23           INC    HL         Null am Zeilenende übergehen
E9F2  F1           POP    AF         Aufrufadresse löschen
E9F3  C3 93 DD     JP     DD93       zur Interpreterschleife

*********************************** Basic-Befehl LOAD
E9F6  CD 0D EA     CALL   EA0D       1. Block lesen und auswerten
E9F9  30 06        JR     NC,EA01    Binärdatei ?
E9FB  CD A8 EB     CALL   EBA8       sonst Basic-Programm laden
E9FE  C3 64 C0     JP     C064       zur Eingabeschleife

EA01  E5           PUSH   HL         Basic-PC retten
EA02  CD 01 F5     CALL   F501       Platz für Binärdatei prüfen
EA05  CD 30 EA     CALL   EA30       Binärdatei laden
EA08  CA 6B CB     JP     Z,CB6B     Abbruch ? dann "Break"
EA0B  E1           POP    HL         Basic-PC zurück
EA0C  C9           RET

*********************************** 1. Block des Prg. lesen/auswerten
                                    OUT: CY=1 für Basic-Programm
                                           CY=0 für Binärdatei
                                           DE: Startadresse
                                           BC: Länge
EA0D  CD 8F EB     CALL   EB8F       1. Block lesen
EA10  E6 0E        AND    0E         Bit für gesch. File löschen
EA12  EE 02        XOR    02         Code für Binärdatei ?
EA14  28 0B        JR     Z,EA21     dann behandeln
EA16  CD 4A DD     CALL   DD4A       auf Statementende prüfen
EA19  CD 8C C1     CALL   C18C       Variablen löschen
```

```
EA1C  CD 6B C1     CALL   C16B         Programm löschen, div. Init.
EA1F  37           SCF                 CY=1 für Basic-Programm
EA20  C9           RET

EA21  CD 55 DD     CALL   DD55         folgt Komma ?
EA24  DC 91 CE     CALL   C,CE91       dann Startadresse holen
EA27  ED 53 3F AE  LD     (AE3F),DE    Startadresse speichern
EA2B  CD 4A DD     CALL   DD4A         auf Statementende prüfen
EA2E  B7           OR     A            CY=0 für Binärdatei
EA2F  C9           RET

*********************************** Binärdatei laden
                                    OUT: Z=1 für Abbruch
                                         CY=0 für Fehler
                                         HL: Aufrufadresse
EA30  2A 3F AE     LD     HL,(AE3F)    Startadresse für Binärdatei
EA33  CD 83 BC     CALL   BC83         CAS IN DIRECT
EA36  E5           PUSH   HL
EA37  DC 7A BC     CALL   C,BC7A       kein Fehler ? d. CAS IN CLOSE
EA3A  E1           POP    HL
EA3B  C9           RET

*********************************** Basic-Befehl CHAIN
EA3C  EE AB        XOR    AB           folgt Token für MERGE ?
EA3E  20 04        JR     NZ,EA44      nein ? dann CHAIN
EA40  CD 3F DD     CALL   DD3F         MERGE-Token übergehen
EA43  37           SCF                 CY=1 für CHAIN MERGE
EA44  9F           SBC    A            0 f. CHAIN, $FF f. CHAIN MERGE
EA45  32 41 AE     LD     (AE41),A     Flag speichern
EA48  CD 8F EB     CALL   EB8F         1. Block lesen
EA4B  11 00 00     LD     DE,0000      Default-Startzeilennr.
EA4E  CD 55 DD     CALL   DD55         folgt Komma ?
EA51  30 06        JR     NC,EA59      nein ? d. Default (Prg.-Start)
EA53  7E           LD     A,(HL)       nächstes Zeichen
EA54  FE 2C        CP     2C           zweites Komma ?
EA56  C4 91 CE     CALL   NZ,CE91      nein ? d. Startzeilennr. holen
EA59  D5           PUSH   DE           Startzeilennr. retten
EA5A  CD 55 DD     CALL   DD55         folgt Komma ?
EA5D  3E 00        LD     A,00         Flag für kein DELETE
EA5F  30 09        JR     NC,EA6A      kein Komma ?
EA61  CD 37 DD     CALL   DD37         sonst Test auf DELETE
EA64  92                               Token für DELETE
EA65  CD 37 E7     CALL   E737         zu löschenden Bereich holen
EA68  3E FF        LD     A,FF         Flag für DELETE
EA6A  F5           PUSH   AF           DELETE-Flag retten
EA6B  CD 4A DD     CALL   DD4A         auf Statementende prüfen
EA6E  CD 1B FB     CALL   FB1B         Strings in Stringb. forcieren
EA71  CD 3E FC     CALL   FC3E         Garbage collection
EA74  CD 89 E9     CALL   E989         Variablenoffsets löschen
EA77  CD D2 D5     CALL   D5D2         definierte Funktionen löschen
EA7A  CD 49 F5     CALL   F549         Variablen in Stringber. retten
EA7D  F1           POP    AF           DELETE-Flag
EA7E  C5           PUSH   BC           Länge der Variablen
EA7F  D5           PUSH   DE           Länge der einfachen Variablen
EA80  B7           OR     A            DELETE-Flag gesetzt ?
EA81  C4 5A E7     CALL   NZ,E75A      dann Bereich löschen
EA84  3A 41 AE     LD     A,(AE41)     Flag für CHAIN/CHAIN MERGE
EA87  B7           OR     A            CHAIN MERGE ?
```

```
EA88  20 08      JR    NZ,EA92      dann Programm mergen
EA8A  CD 6B C1   CALL  C16B         Programm löschen
EA8D  CD A8 EB   CALL  EBA8         Programm laden
EA90  18 03      JR    EA95
EA92  CD 9D EB   CALL  EB9D         Programm mergen
EA95  D1         POP   DE           Länge der einfachen Variablen
EA96  C1         POP   BC           Länge der Variablen
EA97  CD 71 F5   CALL  F571         Var. aus Stringbereich zurück
EA9A  D1         POP   DE           Startzeilennummer
EA9B  2A 81 AE   LD    HL,(AE81)    Zeiger auf Programmstart
EA9E  7A         LD    A,D            Flag für Programmstart ?
EA9F  B3         OR    E
EAA0  C8         RET   Z            dann fertig
EAA1  CD 9A E7   CALL  E79A         sonst Zeilenadresse holen
EAA4  2B         DEC   HL           Zeiger auf Zeilenende davor
EAA5  C9         RET

********************************** Basic-Befehl MERGE
EAA6  CD 8F EB   CALL  EB8F           1. Block lesen
EAA9  CD 4A DD   CALL  DD4A         auf Statemententende prüfen
EAAC  CD 8C C1   CALL  C18C         Variablen löschen
EAAF  CD 9D EB   CALL  EB9D         Programm mergen
EAB2  C3 64 C0   JP    C064         zur Eingabeschleife

********************************** Programm mergen
EAB5  CD 7A C1   CALL  C17A         Basic-Zeiger initialisieren
EAB8  CD 87 E6   CALL  E687         Zeilenadresse eliminieren
EABB  2A 83 AE   LD    HL,(AE83)    Zeiger auf Programmende
EABE  EB         EX    DE,HL        nach DE
EABF  2A 81 AE   LD    HL,(AE81)    Zeiger auf Programmstart
EAC2  23         INC   HL           Zeiger auf erste Zeile
EAC3  22 83 AE   LD    (AE83),HL    als neues Programmende setzen
EAC6  EB         EX    DE,HL          altes Programmende
EAC7  CD DA FF   CALL  FFDA           -Programmstart
EACA  EB         EX    DE,HL          =Programmlänge, nach BC
EACB  2A 8D B0   LD    HL,(B08D)    Zeiger auf Start der Strings
EACE  EB         EX    DE,HL        als Zieladresse nach DE
EACF  2B         DEC   HL           Zeiger a. letztes Programmbyte
EAD0  CD F5 FF   CALL  FFF5         Programm nach oben verschieben
EAD3  13         INC   DE           Zeiger auf erste Zeile
EAD4  EB         EX    DE,HL        nach HL
EAD5  E5         PUSH  HL           Zeiger auf Zeile d. alten Prg.
EAD6  2A 83 AE   LD    HL,(AE83)    Zeiger auf neues Programmende
EAD9  11 20 00   LD    DE,0020      min. Platz
EADC  19         ADD   HL,DE        addieren
EADD  EB         EX    DE,HL        Endzeiger nach DE
EADE  E1         POP   HL           Zeiger auf Zeile d. alten Prg.
EADF  CD B8 FF   CALL  FFB8         mit Endzeiger vergleichen
EAE2  38 50      JR    C,EB34       kein Platz ? dann Fehler
EAE4  CD 84 EB   CALL  EB84         näch. Zeilenlänge von Kassette
EAE7  B3         OR    E              Zeilenlänge =0 ?
EAE8  28 30      JR    Z,EB1A       dann Programmende
EAEA  D5         PUSH  DE           Zeilenlänge
EAEB  CD 84 EB   CALL  EB84         Zeilennr. von Kassette holen
EAEE  E5         PUSH  HL           Zeiger auf Zeile d. alten Prg.
EAEF  7E         LD    A,(HL)
EAF0  23         INC   HL             nächste Zeilenlänge =0 ?
EAF1  B6         OR    (HL)
```

```
EAF2  28 12     JR    Z,EB06       dann altes Programm zu Ende
EAF4  23        INC   HL
EAF5  7E        LD    A,(HL)         nächste Zeilennr.
EAF6  23        INC   HL             aus altem Programm
EAF7  66        LD    H,(HL)
EAF8  6F        LD    L,A
EAF9  CD B8 FF  CALL  FFB8         m. zu ladender Nr. vergleichen
EAFC  E1        POP   HL           Zeiger auf Zeile d. alten Prg.
EAFD  28 0F     JR    Z,EB0E       gleich ? d. alte Zeile überg.
EAFF  30 06     JR    NC,EB07      Nr. aus altem Prg. größer ?
EB01  CD 48 EB  CALL  EB48         Zeile aus altem Programm kop.
EB04  18 E8     JR    EAEE         nächste Zeile des alten Prg.
EB06  E1        POP   HL             zweitoberstes Stackelement
EB07  E3        EX    (SP),HL        (Länge der neuen Zeile)
EB08  CD 5E EB  CALL  EB5E         Zeile von Kassette laden
EB0B  E1        POP   HL           Zeiger auf Zeile d. alten Prg.
EB0C  18 C7     JR    EAD5         weiter laden
EB0E  E3        EX    (SP),HL      Zg. alte Zeile r., Länge zur.
EB0F  CD 5E EB  CALL  EB5E         Zeile von Kassette laden
EB12  E1        POP   HL           Zeiger auf Zeile d. alten Prg.
EB13  5E        LD    E,(HL)
EB14  23        INC   HL             Zeilenlänge laden
EB15  56        LD    D,(HL)
EB16  2B        DEC   HL           Zeiger auf Zeile
EB17  19        ADD   HL,DE        Länge addieren
EB18  18 BB     JR    EAD5         weiter laden
EB1A  7E        LD    A,(HL)
EB1B  23        INC   HL             nächste Zeilenlänge
EB1C  B6        OR    (HL)
EB1D  2B        DEC   HL
EB1E  28 05     JR    Z,EB25       altes Programm zu Ende ?
EB20  CD 48 EB  CALL  EB48         Zeile aus altem Prg. kopieren
EB23  18 F5     JR    EB1A         altes Prg. weiter kopieren
EB25  2A 83 AE  LD    HL,(AE83)    Zeiger auf Programmende
EB28  36 00     LD    (HL),00
EB2A  23        INC   HL             Zeilenlänge Null
EB2B  36 00     LD    (HL),00        als Endkennzeichen
EB2D  23        INC   HL
EB2E  22 83 AE  LD    (AE83),HL    Zeiger auf Programmende setzen
EB31  C3 B1 D5  JP    D5B1         Variablenbereich freigeben

EB34  1E 07     LD    E,07         Nr. für "Memory full"
EB36  18 02     JR    EB3A
EB38  1E 18     LD    E,18         Nr. für "EOF met"
EB3A  D5        PUSH  DE           Fehlernr. retten
EB3B  CD AD D2  CALL  D2AD         Kassette init./abbrechen
EB3E  CD 8C C1  CALL  C18C         Variablen löschen
EB41  CD 6B C1  CALL  C16B         Programm löschen
EB44  D1        POP   DE           Fehlernr. zurück
EB45  C3 94 CA  JP    CA94         Fehler ausgeben

************************************ Zeile aus altem Programm kopieren
                                   IN : HL: Zeiger auf zu kop. Zeile
                                   OUT: HL: Zeiger nach Zeile

EB48  C5        PUSH  BC
EB49  D5        PUSH  DE
EB4A  E5        PUSH  HL           Zeiger auf Zeile d. alten Prg.
EB4B  4E        LD    C,(HL)
```

```
EB4C  23        INC   HL               Zeilenlänge laden
EB4D  46        LD    B,(HL)
EB4E  2A 83 AE  LD    HL,(AE83)        bisheriges Programmende des
EB51  EB        EX    DE,HL            neuen Programms nach DE
EB52  E1        POP   HL             Zeiger auf Zeile d. alten Prg.
EB53  CD F2 FF  CALL  FFF2           Zeile in neues Prg. kopieren
EB56  EB        EX    DE,HL            Endadresse der Zeile
EB57  22 83 AE  LD    (AE83),HL        als neues Programmende
EB5A  EB        EX    DE,HL            setzen
EB5B  D1        POP   DE
EB5C  C1        POP   BC
EB5D  C9        RET

***********************************  Programmzeile von Kassette laden
                                     IN : HL: Zeilenlänge
                                          DE: Zeilennummer
                                          beim CPC 664/6128:
                                     OUT: CY=0 für Fehler (EOF)
EB5E  D5        PUSH  DE             Zeilennr. retten
EB5F  EB        EX    DE,HL          Zeilenlänge nach DE
EB60  2A 83 AE  LD    HL,(AE83)      bisheriges Programmende
EB63  73        LD    (HL),E
EB64  23        INC   HL               Zeilenlänge in folgende
EB65  72        LD    (HL),D           Zeile eintragen
EB66  23        INC   HL
EB67  EB        EX    DE,HL            Zeilenlänge retten,
EB68  E3        EX    (SP),HL          Zeilennummer zurück
EB69  EB        EX    DE,HL
EB6A  73        LD    (HL),E
EB6B  23        INC   HL               Zeilennummer in Zeile
EB6C  72        LD    (HL),D           eintragen
EB6D  23        INC   HL
EB6E  D1        POP   DE             Zeilenlänge
EB6F  1B        DEC   DE
EB70  1B        DEC   DE               4 Bytes für Zeilenlänge
EB71  1B        DEC   DE               und Zeilennr. übergehen
EB72  1B        DEC   DE
EB73  7A        LD    A,D              restliche Zeilenlänge
EB74  B3        OR    E                =0 ?
EB75  28 09     JR    Z,EB80         dann Zeile fertig geladen
EB77  CD 80 BC  CALL  BC80           Zeichen von Kassette
EB7A  30 BC     JR    NC,EB38        EOF ? dann "EOF met"
EB7C  77        LD    (HL),A         sonst Zeichen in Zeile speich.
EB7D  23        INC   HL
EB7E  18 F2     JR    EB72           Zeile weiter laden
EB80  22 83 AE  LD    (AE83),HL      neues Programmende setzen
EB83  C9        RET

***********************************  Zwei-Byte-Wert von Kassette laden
                                     OUT: DE: Wert; A: Hi-Byte
                                          beim CPC 664/6128:
                                     OUT: CY=0 für Fehler (EOF)
EB84  CD 80 BC  CALL  BC80           Zeichen von Kassette holen
EB87  5F        LD    E,A            als Lo-Byte
EB88  DC 80 BC  CALL  C,BC80         kein EOF ? dann 2. Zeichen
EB8B  30 AB     JR    NC,EB38        EOF ? dann "EOF met"
EB8D  57        LD    D,A            2. Zeichen als Hi-Byte
EB8E  C9        RET
```

```
***********************************   1. Block des Programms lesen
                                      OUT: A: Filetyp
                                           BC: Länge
                                           DE: Startadresse
EB8F  CD AD D2     CALL   D2AD           Kassette initialisieren
EB92  CD 6A D2     CALL   D26A           Eingabefile öffnen
EB95  32 42 AE     LD     (AE42),A       Filetyp
EB98  ED 43 43 AE  LD     (AE43),BC      und Länge speichern
EB9C  C9           RET

***********************************   norm. bzw. ASCII-Programm mergen
EB9D  3A 42 AE     LD     A,(AE42)       Filetyp
EBA0  B7           OR     A              ungeschütztes Programm ?
EBA1  CA B5 EA     JP     Z,EAB5         dann mergen
EBA4  FE 16        CP     16             ASCII-Datei ?
EBA6  20 0B        JR     NZ,EBB3        nein ? dann Fehler

***********************************   norm. bzw. ASCII-Programm laden
EBA8  3A 42 AE     LD     A,(AE42)       Filetyp
EBAB  FE 16        CP     16             ASCII-Datei ?
EBAD  28 40        JR     Z,EBEF         dann laden
EBAF  E6 FE        AND    FE             Flag für geschützt löschen
EBB1  28 05        JR     Z,EBB8         Basic-Programm ? dann laden
EBB3  1E 19        LD     E,19           Nr. für "File type error"
EBB5  C3 94 CA     JP     CA94           Fehler ausgeben
EBB8  CD 7A C1     CALL   C17A           Basic-Zeiger initialisieren
EBBB  2A 81 AE     LD     HL,(AE81)      Zeiger auf Programmstart
EBBE  23           INC    HL             Zeiger auf erste Zeile
EBBF  EB           EX     DE,HL          nach DE
EBC0  2A 8D B0     LD     HL,(B08D)      Zeiger auf Start der Strings
EBC3  01 80 FF     LD     BC,FF80        -$80 (min. Platz)
EBC6  09           ADD    HL,BC          addieren
EBC7  ED 4B 43 AE  LD     BC,(AE43)      Progammlänge
EBCB  CD CF FF     CALL   FFCF           -Prg.-Start, gibt freien Platz
EBCE  D4 BE FF     CALL   NC,FFBE        ggf. mit Prg.-Länge vergleich.
EBD1  DA 34 EB     JP     C,EB34         zu lang ? dann Fehler
EBD4  60           LD     H,B              Programmlänge
EBD5  69           LD     L,C              nach HL
EBD6  19           ADD    HL,DE          zu Zeiger auf 1. Zeile add.
EBD7  22 83 AE     LD     (AE83),HL      als neues Programmende setzen
EBDA  3A 42 AE     LD     A,(AE42)       Filetyp
EBDD  1F           RRA                     $FF, wenn geschützt,
EBDE  9F           SBC    A                0, wenn ungeschützt
EBDF  32 45 AE     LD     (AE45),A       Flag f. geschützt. Prg. setzen
EBE2  EB           EX     DE,HL          Zeiger auf 1. Zeile
EBE3  CD 83 BC     CALL   BC83           CAS IN DIRECT, Programm laden
EBE6  CA 38 EB     JP     Z,EB38         EOF ? dann "EOF met"
EBE9  CD B1 D5     CALL   D5B1           Variablenbereich freigeben
EBEC  C3 98 D2     JP     D298           Eingabefile schließen

***********************************   ASCII-Programm laden
EBEF  CD 7A C1     CALL   C17A           Basic-Zeiger initialisieren
EBF2  CD CB DD     CALL   DDCB           Direkt-Modus einschalten
EBF5  CD 4C CA     CALL   CA4C           Zeile von Kassette laden
EBF8  D2 98 D2     JP     NC,D298        EOF ? dann CLOSEIN
EBFB  CD BC E6     CALL   E6BC           Zeile auswerten, ggf. einfügen
EBFE  38 F5        JR     C,EBF5         Programmzeile ? dann weiter
EC00  1E 15        LD     E,15           Nr. für "Direct command found"
```

```
EC02  28 02       JR     Z,EC06        Direkteingabe ?
EC04  1E 06       LD     E,06          sonst Nr. für "Overflow"
EC06  C3 94 CA    JP     CA94          Fehler ausgeben

********************************** Basic-Befehl SAVE
EC09  CD AD D2    CALL   D2AD          Kassette initialisieren
EC0C  CD 56 D2    CALL   D256          Ausgabefile öffnen
EC0F  06 00       LD     B,00          Typ für ungeschütztes Programm
EC11  CD 55 DD    CALL   DD55          folgt Komma ?
EC14  30 29       JR     NC,EC3F       nein ? dann speichern
EC16  CD 37 DD    CALL   DD37          Test auf unmarkierte Variable
EC19  0D                               Token für unmarkierte Variable
EC1A  23          INC    HL              Variablenoffset
EC1B  23          INC    HL              übergehen
EC1C  7E          LD     A,(HL)        1. Byte des Namens
EC1D  23          INC    HL            Zeiger nach Namen
EC1E  E6 DF       AND    DF            auf Großschrift forcieren
EC20  F2 38 EC    JP     P,EC38        Name zu Ende ? sonst Fehler
EC23  E5          PUSH   HL            Basic-PC retten
EC24  21 2C EC    LD     HL,EC2C       Zeiger auf Tabelle
EC27  CD 93 FF    CALL   FF93          Adresse entspr. Token holen
EC2A  E3          EX     (SP),HL       Adresse auf Stack, PC zurück
EC2B  C9          RET                  entspr. Routine anspringen

********************************** Tabelle für SAVE
EC2C  03                               3 Tabelleneinträge
EC2D  38 EC                            EC38, "Syntax error" als Def.
EC2F  C1                               "A"+$80
EC30  87 EC                            SAVE ,A
EC32  C2                               "B"+$80
EC33  5C EC                            SAVE ,B
EC35  D0                               "P"+$80
EC36  3D EC                            SAVE ,P

EC38  1E 02       LD     E,02          Nr. für "Syntax error"
EC3A  C3 94 CA    JP     CA94          Fehler ausgeben

********************************** SAVE ,P
EC3D  06 01       LD     B,01          Typ für ungeschütztes Programm
EC3F  CD 4A DD    CALL   DD4A          auf Statementende prüfen
EC42  E5          PUSH   HL            Basic-PC
EC43  C5          PUSH   BC            und Programmtyp retten
EC44  CD 87 E6    CALL   E687          Zeilenadressen eliminieren
EC47  CD 89 E9    CALL   E989          Variablenoffsets löschen
EC4A  2A 81 AE    LD     HL,(AE81)     Zeiger auf Programmstart
EC4D  23          INC    HL            Zeiger auf 1. Zeile
EC4E  EB          EX     DE,HL         nach DE
EC4F  2A 83 AE    LD     HL,(AE83)     Zeiger auf Programmende
EC52  CD CF FF    CALL   FFCF          Start subtrahieren, gibt Länge
EC55  EB          EX     DE,HL         Länge nach DE, Startadr. n. HL
EC56  F1          POP    AF            Programmtyp
EC57  01 00 00    LD     BC,0000       Aufrufadresse
EC5A  18 23       JR     EC7F          Programm speichern

********************************** SAVE ,B
EC5C  06 02       LD     B,02          Typ für Binärdatei
EC5E  CD 37 DD    CALL   DD37          Test auf Komma
EC61  2C                               ","
```

```
EC62  CD 91 CE    CALL   CE91        Startadresse holen
EC65  D5          PUSH   DE          und retten
EC66  CD 37 DD    CALL   DD37        Test auf Komma
EC69  2C                             ","
EC6A  CD 91 CE    CALL   CE91        Länge holen
EC6D  D5          PUSH   DE          und retten
EC6E  CD 55 DD    CALL   DD55        folgt Komma ?
EC71  11 00 00    LD     DE,0000     Default-Aufrufadresse
EC74  DC 91 CE    CALL   C,CE91      kein Komma ? d. Adresse holen
EC77  D5          PUSH   DE          Aufrufadresse retten
EC78  CD 4A DD    CALL   DD4A        auf Statementende prüfen
EC7B  78          LD     A,B         Programmtyp
EC7C  C1          POP    BC          Aufrufadresse
EC7D  D1          POP    DE          und Länge vom Stack
EC7E  E3          EX     (SP),HL     PC retten, Startadr. zurück
EC7F  CD 98 BC    CALL   BC98        CAS OUT DIRECT, Prg. speichern
EC82  D2 6B CB    JP     NC,CB6B     Abbruch ? dann "Break"
EC85  18 17       JR     EC9E        Ausgabefile schließen, zurück

**********************************  SAVE ,A
EC87  CD 4A DD    CALL   DD4A        auf Statementende prüfen
EC8A  E5          PUSH   HL          Basic-PC retten
EC8B  3E 09       LD     A,09        Nr. für Kassette
EC8D  CD A2 C1    CALL   C1A2        als akt. Streamnr. setzen
EC90  F5          PUSH   AF          alte Streamnr. retten
EC91  01 01 00    LD     BC,0001     Startzeilennr.
EC94  11 FF FF    LD     DE,FFFF     Endzeilennr.
EC97  CD 0D E1    CALL   E10D        Programm auf Kassette listen
EC9A  F1          POP    AF          alte Streamnr.
EC9B  CD A2 C1    CALL   C1A2        wieder setzen
EC9E  CD A1 D2    CALL   D2A1        Ausgabefile schließen
ECA1  E1          POP    HL          Basic-PC zurück
ECA2  C9          RET

**********************************  ASCII nach binär wandeln
                                    IN : HL: Zeiger auf String
                                    OUT: Zahl im FAC
                                         A: Zahlenbasis
                                         HL: Zeiger nach String
                                         CY=0, wenn Überlauf
ECA3  CD 44 ED    CALL   ED44        Vorzeichen d. Zahl feststellen
ECA6  20 05       JR     NZ,ECAD     kein Vorzeichen ?
ECA8  CD 61 DD    CALL   DD61        Spaces, TABs und LFs überlesen
ECAB  18 2F       JR     ECDC        und Dezimalstring wandeln
ECAD  FE 26       CP     26          "&" ?
ECAF  28 1C       JR     Z,ECCD      dann Hex-/Binär-String wandeln
ECB1  CD 7F FF    CALL   FF7F        Ziffer oder "." ?
ECB4  38 26       JR     C,ECDC      dann Dezimalstring wandeln
ECB6  CD 10 FF    CALL   FF10        Integer als FAC-Typ
ECB9  CD F3 FE    CALL   FEF3        FAC löschen
ECBC  37          SCF                CY=1 für o.k.
ECBD  C9          RET

**********************************  String in positive Binärzahl
                                    IN : HL: Zeiger auf String
                                    OUT: A: Zahlenbasis
                                         CY=1, wenn o.k.
                                         HL: Zeiger nach String
```

```
                                            DE: Zeiger auf String
                                            Zahl im FAC
                                            CY=0, wenn Überlauf
                                            HL: Zeiger auf String
                                            DE: Zeiger nach String
ECBE  E5          PUSH   HL               Eingabe-Zeiger retten
ECBF  CD C6 EC    CALL   ECC6             String wandeln
ECC2  D1          POP    DE               Zeiger auf Ziffernstring
ECC3  D8          RET    C                kein Fehler ?
ECC4  EB          EX     DE,HL            sonst Zeiger vertauschen
ECC5  C9          RET

**********************************     String in positive Binärzahl
                                       IN : HL: Zeiger auf String
                                       OUT: A: Zahlenbasis
                                            CY=0, wenn Überlauf
                                            HL: Zeiger nach String
                                            DE: Zeiger auf String
                                            Zahl im FAC
ECC6  16 00       LD     D,00             Vorzeichen positiv
ECC8  7E          LD     A,(HL)           1. Zeichen
ECC9  FE 26       CP     26               "&" ?
ECCB  20 0F       JR     NZ,ECDC          nein ? dann Dezimalstring

**********************************     Hex-/Binär-String nach Integer
                                       IN : HL: Zeiger auf String
                                       OUT: Zahl im FAC
                                            A: Zahlenbasis
                                            HL: Zeiger nach String
                                            CY=0, wenn Überlauf
ECCD  CD 1C EE    CALL   EE1C             String nach Integer wandeln
ECD0  EB          EX     DE,HL            Zahl n. HL, Zeiger nach DE
ECD1  F5          PUSH   AF               Zahlenbasis retten
ECD2  CD 0D FF    CALL   FF0D             Zahl in FAC eintragen
ECD5  F1          POP    AF               Zahlenbasis
ECD6  EB          EX     DE,HL            Zeiger nach Zahl nach HL
ECD7  D8          RET    C                kein Fehler ?
ECD8  C8          RET    Z                keine Ziffern ?
ECD9  C3 F3 CA    JP     CAF3             sonst "Overflow", CY=0

**********************************     Dezimalstring nach Integer/REAL
                                       IN : HL: Zeiger auf String
                                            D: Vorzeichen
                                       OUT: HL: Zeiger nach String
                                            A: Zahlenbasis 10
                                            Zahl im FAC
                                            CY=0, wenn Überlauf
ECDC  E5          PUSH   HL               Zeiger auf Ziffernstring
ECDD  7E          LD     A,(HL)           1. Zeichen
ECDE  23          INC    HL               Eingabezeiger erhöhen
ECDF  FE 2E       CP     2E               "." ?
ECE1  CC 61 DD    CALL   Z,DD61           dann Spaces, TABs, LFs überl.
ECE4  CD 83 FF    CALL   FF83             Test auf Ziffer
ECE7  E1          POP    HL               Zeiger wieder auf Anfang
ECE8  38 06       JR     C,ECF0           Ziffer ? dann auswerten
ECEA  7E          LD     A,(HL)           1. Zeichen
ECEB  EE 2E       XOR    2E               "." ?
ECED  C0          RET    NZ               nein ?
```

```
ECEE  23          INC   HL          Zeiger auf Zeichen danach
ECEF  C9          RET
ECF0  CD 10 FF    CALL  FF10        FAC-Typ auf Integer
ECF3  D5          PUSH  DE          Vorzeichen retten
ECF4  01 00 00    LD    BC,0000     Zähler f. Stellen/Nachkommast.
ECF7  11 46 AE    LD    DE,AE46     Zeiger auf Buffer f. Wandlung
ECFA  CD 53 ED    CALL  ED53        Vorkomma-Ziff. n. unpacked BCD
ECFD  FE 2E       CP    2E          "." ?
ECFF  20 0B       JR    NZ,ED0C     keine Nachkommastellen ?
ED01  CD C9 ED    CALL  EDC9        nächstes Zeichen
ED04  CD 19 FF    CALL  FF19        FAC-Typ auf REAL setzen
ED07  0C          INC   C           Flag f. Nachkommastellen setz.
ED08  CD 53 ED    CALL  ED53        Nachkomma-Ziffern n. unp. BCD
ED0B  0D          DEC   C           Zahl d. Nachkommast. korrig.
ED0C  F5          PUSH  AF          Zeichen nach Ziffernstring
ED0D  3E FF       LD    A,FF          $FF als Kennzeichen
ED0F  12          LD    (DE),A        für BCD-Buffer-Ende
ED10  F1          POP   AF          Zeichen nach Ziffernstring
ED11  CD 77 ED    CALL  ED77        dez. Exp. d. letzten Stelle h.
ED14  D1          POP   DE          Vorzeichen
ED15  5F          LD    E,A         Exponent nach E
ED16  E5          PUSH  HL          Eingabezeiger
ED17  D5          PUSH  DE          Vorzeichen/Exponenten retten
ED18  21 46 AE    LD    HL,AE46     Zeiger auf BCD-Buffer
ED1B  CD CE ED    CALL  EDCE        unpacked BCD nach Binärzahl
ED1E  D1          POP   DE          Vorzeichen/Exponenten zurück
ED1F  CD 27 FF    CALL  FF27        Typflag des FAC holen
ED22  30 08       JR    NC,ED2C     REAL-Zahl ?
ED24  E5          PUSH  HL          Zeiger auf Binärzahl
ED25  42          LD    B,D         Vorzeichen
ED26  CD 06 FE    CALL  FE06        setzen, Integer nach FAC
ED29  E1          POP   HL          Zeiger auf Binärzahl
ED2A  38 11       JR    C,ED3D      Zahl nicht zu groß f. Int. ?
ED2C  7A          LD    A,D         sonst Vorzeichen der Zahl
ED2D  4E          LD    C,(HL)      1. Byte aus Binärzahl
ED2E  23          INC   HL          Zeiger auf 2. Byte
ED2F  CD 94 BD    CALL  BD94        5-Byte-Integer nach REAL
ED32  7B          LD    A,E         dez. Exponent
ED33  CD 55 BD    CALL  BD55        Zahl mit 10^A multiplizieren
ED36  EB          EX    DE,HL       Zeiger auf REAL-Zahl nach DE
ED37  CD 16 FF    CALL  FF16        FAC-Typ auf REAL, Zg. n. HL
ED3A  DC 3D BD    CALL  C,BD3D      kein Fehler ? d. REAL n. FAC
ED3D  3E 0A       LD    A,0A        Zahlenbasis =10
ED3F  E1          POP   HL          Eingabezeiger zurück
ED40  D8          RET   C           kein Übertrag ?
ED41  C3 F3 CA    JP    CAF3        "Overflow" ausgeben, CY=0

***********************************   Vorzeichen im String bestimmen
                                      IN : HL: Zeiger auf String
                                      OUT: Z=0, wenn kein Vorzeichen
                                           D: Vorzeichen der Zahl
                                           HL: Zeiger auf Ziffern
ED44  CD 61 DD    CALL  DD61        Spaces, TABs und LFs überlesen
ED47  23          INC   HL          Zeiger auf 1. Ziffer
ED48  16 FF       LD    D,FF        Flag für negatives Vorzeichen
ED4A  FE 2D       CP    2D          "-" ?
ED4C  C8          RET   Z           dann negativ
ED4D  14          INC   D           Flag für positives Vorzeichen
```

```
ED4E  FE 2B       CP    2B          "+" ?
ED50  C8          RET   Z           dann positiv
ED51  2B          DEC   HL          Zeiger auf 1. Ziffer, positiv
ED52  C9          RET

**********************************  Ziffernstring nach unpacked BCD
                                    IN/OUT: HL: Eingabezeiger
                                            DE: BCD-Bufferzeiger
                                            B: Gesamtstellenzahl
                                            C: Nachkommastellen +1
                                            C=0, wenn k. Nachk.-St.
ED53  E5          PUSH  HL            Eingabezeiger retten
ED54  CD 61 DD    CALL  DD61          Spaces, TABs und LFs überlesen
ED57  23          INC   HL            Zeiger auf nächstes Zeichen
ED58  CD 83 FF    CALL  FF83          Test auf Ziffer
ED5B  38 04       JR    C,ED61        Ziffer ?
ED5D  E1          POP   HL            sonst Eingabezeiger zurück
ED5E  C3 8A FF    JP    FF8A          auf Großschrift forcieren
ED61  E3          EX    (SP),HL         Eintrag vom
ED62  E1          POP   HL              Stack löschen
ED63  D6 30       SUB   30            Ziffernwert berechnen
ED65  12          LD    (DE),A        in Buffer speichern
ED66  B0          OR    B             führende Null ?
ED67  28 07       JR    Z,ED70        dann unterdrücken
ED69  78          LD    A,B           Gesamtstellenzahl
ED6A  04          INC   B             erhöhen
ED6B  FE 0C       CP    0C            schon max. Ziffernzahl ?
ED6D  30 01       JR    NC,ED70       dann nicht weiter speichern
ED6F  13          INC   DE            Bufferzeiger erhöhen
ED70  79          LD    A,C           Zahl der Nachkommastellen
ED71  B7          OR    A             keine Nachkommastellen ?
ED72  28 DF       JR    Z,ED53        dann nächste Ziffer
ED74  0C          INC   C             sonst Nachkommastellenz. erh.
ED75  18 DC       JR    ED53          nächste Ziffer

**********************************  dez. Exponenten holen/berechnen
                                    IN : A: 1. Zeichen
                                         HL: Eingabezeiger
                                         B: Gesamtstellenzahl
                                         C: Nachkommastellenzahl
                                    OUT: A: Exponent der letzten
                                            Bufferstelle
ED77  FE 45       CP    45            "E" ?
ED79  20 10       JR    NZ,ED8B       dann Exponent zunächst =0
ED7B  E5          PUSH  HL            Eingabezeiger retten
ED7C  CD C9 ED    CALL  EDC9          nächstes Zeichen holen
ED7F  CD 44 ED    CALL  ED44          Vorzeichen des Exp. nach D
ED82  CC 61 DD    CALL  Z,DD61        kein Vorz. ? d. " "/TAB/LF üb.
ED85  CD 83 FF    CALL  FF83          Test auf Ziffer
ED88  38 04       JR    C,ED8E        Ziffer ? dann Exp. holen
ED8A  E1          POP   HL            sonst Zeiger wieder auf Anfang
ED8B  AF          XOR   A             Exponent =0
ED8C  18 1E       JR    EDAC          endgültigen Exp. berechnen
ED8E  E3          EX    (SP),HL         obersten Stackeintrag
ED8F  E1          POP   HL              löschen
ED90  CD 19 FF    CALL  FF19          FAC-Typ auf REAL
ED93  D5          PUSH  DE            Vorzeichen
ED94  C5          PUSH  BC            Gesamt/Nachk.-Stellenzahl
```

```
ED95  CD 35 EE    CALL  EE35       Ziffern nach binär, nach DE
ED98  30 09       JR    NC,EDA3    Fehler ?
ED9A  7B          LD    A,E        Lo-Byte des Exponenten
ED9B  D6 64       SUB   64         minus 100
ED9D  7A          LD    A,D          Übertrag auf Hi-Byte
ED9E  DE 00       SBC   00           berücksichtigen
EDA0  7B          LD    A,E        Lo-Byte
EDA1  38 02       JR    C,EDA5     Exponent <100 ?
EDA3  3E 7F       LD    A,7F       sonst max. Wert
EDA5  C1          POP   BC         Gesamt/Nachk.-Stellenzahl
EDA6  D1          POP   DE         und Vorzeichen zurück
EDA7  14          INC   D          Vorzeichen des Exponenten
EDA8  20 02       JR    NZ,EDAC    positiv ?
EDAA  2F          CPL                sonst Zweierkomplement
EDAB  3C          INC   A            bilden
EDAC  C6 80       ADD   80         nach FAC-Speicherweise
EDAE  5F          LD    E,A
EDAF  78          LD    A,B        Gesamtstellenzahl
EDB0  D6 0C       SUB   0C         -12 =Z. d. St. außerh. Buffer
EDB2  30 01       JR    NC,EDB5    nicht alle Stellen im Buffer ?
EDB4  AF          XOR   A          keine Stelle nicht in Buffer
EDB5  91          SUB   C          - Nachkommastellenzahl
EDB6  30 09       JR    NC,EDC1    auch Vork.-St. n. im Buffer ?
EDB8  83          ADD   E          Exponent addieren
EDB9  38 01       JR    C,EDBC     Exp. groß genug ?
EDBB  AF          XOR   A          sonst min. Exp. setzen
EDBC  FE 01       CP    01           Exponenten wieder auf
EDBE  CE 80       ADC   80           Zweierkomplement-Format
EDC0  C9          RET
EDC1  83          ADD   E          Exponent addieren
EDC2  30 02       JR    NC,EDC6    Exp. nicht zu groß ?
EDC4  3E FF       LD    A,FF       sonst max. Exp. setzen
EDC6  D6 80       SUB   80         wieder nach Zweierkomp.-Format
EDC8  C9          RET

*********************************** nächstes Zeichen aus Zahl holen
                                   IN/OUT: HL: Eingabezeiger
                                            A: Zeichen
EDC9  CD 61 DD    CALL  DD61       Spaces, TABs und LFs überlesen
EDCC  23          INC   HL         Zeiger auf nächstes Zeichen
EDCD  C9          RET

*********************************** unpacked BCD nach Binär wandeln
                                   IN : HL: Zeiger auf BCD-Buffer
                                   OUT: HL: Zeiger auf Binärzahl
                                         C: Länge der Binärzahl
EDCE  EB          EX    DE,HL      BCD-Bufferzeiger nach DE
EDCF  21 58 AE    LD    HL,AE58    Zeiger nach Binärzahl-Buffer
EDD2  01 01 05    LD    BC,0501    Bufferlänge/Länge der Zahl
EDD5  2B          DEC   HL           Zahl auf
EDD6  36 00       LD    (HL),00      Null setzen
EDD8  10 FB       DJNZ  EDD5       weitere Bytes zu löschen ?
EDDA  1A          LD    A,(DE)     1. Zeichen aus Ziffernbuffer
EDDB  FE FF       CP    FF         Endekennzeichen ?
EDDD  C8          RET   Z          dann Zahl=0, zurück
EDDE  77          LD    (HL),A     Ziffernwert in Buffer
EDDF  21 53 AE    LD    HL,AE53    Zeiger auf Binärzahl-Buffer
EDE2  13          INC   DE         Zeiger auf nächste Ziffer
```

```
EDE3  1A        LD    A,(DE)      nächster Ziffernwert
EDE4  FE FF     CP    FF          Kennz. f. Bufferende ?
EDE6  C8        RET   Z           dann fertig
EDE7  D5        PUSH  DE          Zeiger auf BCD-Buffer retten
EDE8  41        LD    B,C         Länge des Binärzahl-Buffers
EDE9  16 00     LD    D,00        Ziffer/Übertrag hi =0
EDEB  E5        PUSH  HL          Zeiger in Binärzahl-Buffer
EDEC  5E        LD    E,(HL)      Byte aus Buffer
EDED  62        LD    H,D
EDEE  6B        LD    L,E
EDEF  29        ADD   HL,HL
EDF0  29        ADD   HL,HL         mit 10 multiplizieren
EDF1  19        ADD   HL,DE
EDF2  29        ADD   HL,HL
EDF3  5F        LD    E,A         Ziffer/Übertrag
EDF4  19        ADD   HL,DE       addieren
EDF5  5D        LD    E,L         Summe lo
EDF6  7C        LD    A,H         Summe hi als neuen Übertrag
EDF7  E1        POP   HL          Zeiger in Binärzahl-Buffer
EDF8  73        LD    (HL),E      Byte in Buffer speichern
EDF9  23        INC   HL          Zeiger auf nächstes Byte
EDFA  10 EF     DJNZ  EDEB        weitere Bytes im Buffer ?
EDFC  D1        POP   DE          Zeiger in BCD-Buffer
EDFD  B7        OR    A           kein Übertr. zu nächst. Byte ?
EDFE  28 DF     JR    Z,EDDF      dann nächste Ziffer
EE00  77        LD    (HL),A      sonst Übertrag speichern
EE01  0C        INC   C           Länge der Binärzahl erhöhen
EE02  18 DB     JR    EDDF        nächste Ziffer

***********************************  Zeilennummern-String wandeln
                                     IN : HL: Eingabezeiger
                                     OUT: CY=1, wenn o.k.
                                          HL: Zeiger nach Zeilennr.
                                          Zeilennr. in FAC und DE
                                          CY=0, wenn Fehler
                                          dann:
                                          Z=0, wenn Überlauf
                                          Z=1, wenn keine Ziffer
                                               oder Zahl=0
                                          HL: Zeiger auf Zeilennr.
                                          DE: Zeiger nach Zeilennr.
EE04  C5        PUSH  BC
EE05  E5        PUSH  HL          Zeiger auf Ziffern retten
EE06  CD 35 EE  CALL  EE35        Dezimalziffern wandeln
EE09  EB        EX    DE,HL         gewandelte Zahl (in DE)
EE0A  CD 0D FF  CALL  FF0D          in FAC eintragen
EE0D  EB        EX    DE,HL
EE0E  C1        POP   BC          Zeiger auf Ziffern
EE0F  30 06     JR    NC,EE17     Fehler ?
EE11  7A        LD    A,D
EE12  B3        OR    E             Zahl <>0 ?
EE13  C6 FF     ADD   FF
EE15  38 03     JR    C,EE1A      dann o.k.
EE17  50        LD    D,B           sonst Zeiger auf Ziffern
EE18  59        LD    E,C           nach DE
EE19  EB        EX    DE,HL       nach HL, Zeiger n. Zahl n. DE
EE1A  C1        POP   BC
EE1B  C9        RET
```

```
************************************    Hex-/Binärstring nach Integer
                                        IN : HL: Eingabezeiger auf "&"
                                        OUT: HL: Eingabezeiger
                                             DE: Zahl
                                             A: Zahlenbasis
                                             CY=1, wenn o.k.
                                             CY=0, Z=0, wenn Zahl zu groß
                                             CY=0, Z=1, wenn keine Ziffer
EE1C  23        INC   HL                  Zeiger nach "&"
EE1D  CD 61 DD  CALL  DD61                Spaces, TABs und LFs überlesen
EE20  CD 8A FF  CALL  FF8A                auf Großschrift forcieren
EE23  06 02     LD    B,02                Zahlenbasis für Binärzahl
EE25  FE 58     CP    58                  "X" ?
EE27  28 06     JR    Z,EE2F              dann Binärzahl
EE29  06 10     LD    B,10                Zahlenbasis für Hex-Zahl
EE2B  FE 48     CP    48                  "H" ?
EE2D  20 04     JR    NZ,EE33             nein ? dann sofort wandeln
EE2F  23        INC   HL                  "H" bzw. "X" übergehen
EE30  CD 61 DD  CALL  DD61                Spaces, TABs und LFs überlesen
EE33  18 02     JR    EE37                Ziffern wandeln

************************************    Dezimalstring nach Integer
                                        IN : HL: Eingabezeiger
                                        OUT: HL: Eingabezeiger
                                             DE: Zahl
                                             A: Zahlenbasis (10)
                                             CY=1, wenn o.k.
                                             CY=0, Z=0, wenn Zahl zu groß
                                             CY=0, Z=1, wenn keine Ziffer
EE35  06 0A     LD    B,0A                Zahlenbasis für Dezimalzahl
EE37  EB        EX    DE,HL               Eingabezeiger nach DE
EE38  CD 61 EE  CALL  EE61                ersten Ziffernwert holen
EE3B  26 00     LD    H,00                Ziffernwert hi=0
EE3D  6F        LD    L,A                 Ziffernwert lo
EE3E  30 1E     JR    NC,EE5E             keine Ziffer ? dann Ende
EE40  0E 00     LD    C,00                Flag für keinen Überlauf
EE42  CD 61 EE  CALL  EE61                nächsten Ziffernwert holen
EE45  30 14     JR    NC,EE5B             keine Ziffer ? dann Ende
EE47  D5        PUSH  DE                  Eingabezeiger retten
EE48  16 00     LD    D,00                Ziffer/Basis hi =0
EE4A  5F        LD    E,A                 Ziffernwert lo
EE4B  D5        PUSH  DE                  Ziffernwert retten
EE4C  58        LD    E,B                 Zahlenbasis
EE4D  CD BE BD  CALL  BDBE                mit alter Zahl multiplizieren
EE50  D1        POP   DE                  neuer Ziffernwert
EE51  38 03     JR    C,EE56              Übertrag bei Multiplikation ?
EE53  19        ADD   HL,DE               sonst Ziffernwert addieren
EE54  30 02     JR    NC,EE58             kein Übertrag ?
EE56  0E FF     LD    C,FF                Flag für Überlauf setzen
EE58  D1        POP   DE                  Eingabezeiger
EE59  18 E7     JR    EE42                nächste Ziffer auswerten
EE5B  79        LD    A,C                 Flag für Überlauf
EE5C  FE 01     CP    01                  CY=1 f. o.k., CY=0 f. Überlauf
EE5E  EB        EX    DE,HL               Eingabezg. nach HL, Zahl n. DE
EE5F  78        LD    A,B                 Zahlenbasis
EE60  C9        RET
```

```
***********************************  Ziffernwert berechnen
                                     IN : DE: Eingabezeiger auf Ziffer
                                          B: Zahlenbasis
                                     OUT: DE: Eingabezeiger
                                          CY=1, wenn Ziffer gültig
                                          A: Ziffernwert
                                          CY=0, Z=1, wenn keine Ziffer
EE61  1A          LD    A,(DE)          Ziffer
EE62  13          INC   DE              Zeiger auf nächstes Zeichen
EE63  CD 83 FF    CALL  FF83            Test auf Ziffer von "0".."9"
EE66  38 0A       JR    C,EE72          Ziffer von "0".."9" ?
EE68  CD 8A FF    CALL  FF8A            auf Großschrift forcieren
EE6B  FE 41       CP    41              "A" ?
EE6D  3F          CCF
EE6E  30 05       JR    NC,EE75         weder Buchst. noch Ziffer ?
EE70  D6 07       SUB   07              Differenz "9"+1 zu "A"
EE72  D6 30       SUB   30              von ASCII nach Ziffernwert
EE74  B8          CP    B               kleiner als Zahlenbasis ?
EE75  D8          RET   C               dann o.k.
EE76  1B          DEC   DE              Zeiger wieder auf Zeichen
EE77  AF          XOR   A               CY=0, Z=1 für Ziffer ungültig
EE78  C9          RET

***********************************  positive Integerzahl ausgeben
                                     IN : HL: Zahl
EE79  CD 0D FF    CALL  FF0D            Zahl in FAC speichern
EE7C  CD 82 EE    CALL  EE82            nach ASCII wandeln
EE7F  C3 41 C3    JP    C341            und ausgeben

***********************************  positive Integerzahl nach ASCII
                                     IN : Zahl im FAC (CPC 464)
                                          HL: Zahl (CPC 664/6128)
                                     OUT: HL: Zeiger auf String
EE82  D5          PUSH  DE
EE83  C5          PUSH  BC
EE84  CD C3 FC    CALL  FCC3            Wandlungs-Parameter holen
EE87  AF          XOR   A               Flag f. keine Formatierung
EE88  CD A7 EE    CALL  EEA7            Zahl nach ASCII wandeln
EE8B  23          INC   HL              Vorzeichen übergehen
EE8C  C1          POP   BC
EE8D  D1          POP   DE
EE8E  C9          RET

***********************************  Zahl nach ASCII, kein pos. Vorz.
                                     IN : Zahl im FAC
                                     OUT: HL: Zeiger auf String
EE8F  D5          PUSH  DE
EE90  C5          PUSH  BC
EE91  AF          XOR   A               Flag f. keine Formatierung
EE92  CD 9F EE    CALL  EE9F            Zahl nach ASCII
EE95  C1          POP   BC
EE96  D1          POP   DE
EE97  7E          LD    A,(HL)          1. Zeichen
EE98  FE 20       CP    20              Space ?
EE9A  C0          RET   NZ              nein ? dann kein pos. Vorz.
EE9B  23          INC   HL              sonst Space übergehen
EE9C  C9          RET
```

```
***********************************     Zahl nach ASCII, max. 9 Ziffern
                                        IN : Zahl im FAC
                                        OUT: HL: Zeiger auf String
EE9D  3E 40       LD    A,40               max. 7 Stellen b. Exp.-Darst.

***********************************     Zahl formatiert nach ASCII
                                        IN : Zahl im FAC
                                             A: Formatierungsflags
                                             b7/b6:
                                              00: normale Darstellung
                                              01: max. 7 Mantissenstellen
                                                  bei Exponentialdarst.
                                              10: formatierte Darstellung
                                              11: form. Exponentialdarst.
                                             b5: Flag f. "*" vor Zahl
                                             b4: Flag f. Vorz. nach Zahl
                                             b3: Flag f. pos. Vorz. ="+"
                                             b2: Flag f. "$" vor d. Zahl
                                             beim 664/6128:
                                               b2: Flag f. Währungszeich.
                                               ($AE54): Währungszeichen
                                             b1: Flag f. Komma-Einteilung
                                             b0: Flag f. Formatüberlauf
                                                 (muß anfangs Null sein)
                                             nur bei form. Darst. (b7=1):
                                             H: Vorkommastellenzahl
                                             L: Nachkommastellenzahl
                                        OUT: HL: Zeiger auf String
EE9F  22 6E AE    LD    (AE6E),HL          Vor-/Nachkommast. speichern
EEA2  F5          PUSH  AF                 Formatierungs-Flags
EEA3  CD B3 FC    CALL  FCB3               Mant. norm., Parameter holen
EEA6  F1          POP   AF                 Formatierungs-Flags
EEA7  C5          PUSH  BC                 Vorzeichen retten
EEA8  57          LD    D,A                Formatierungs-Flags
EEA9  D5          PUSH  DE                 und Kommaposition retten
EEAA  EB          EX    DE,HL              Adr. d. höchstwert. Byte n. DE
EEAB  21 68 AE    LD    HL,AE68            Zeiger in Buffer f. Wandlung
EEAE  36 00       LD    (HL),00            Null als Endkennzeichen
EEB0  22 70 AE    LD    (AE70),HL          Buffer-Endzeiger setzen
EEB3  CD B7 F0    CALL  F0B7               Mantisse nach ASCII wandeln
EEB6  D1          POP   DE                 Flags/Kommaposition zurück
EEB7  CD D4 EE    CALL  EED4               Dezimalpunkt und Exp. setzen
EEBA  CD 3D F0    CALL  F03D               Komma-Einteilungen setzen
EEBD  58          LD    E,B                Zahl der Vorkommastellen
EEBE  C1          POP   BC                 Vorzeichen
EEBF  7B          LD    A,E                Zahl der Vorkommastellen
EEC0  B7          OR    A                  keine Vorkommastellen ?
EEC1  CC 50 F0    CALL  Z,F050             dann führende Null setzen
EEC4  CD 5F F0    CALL  F05F               ggf. führendes "$" setzen
EEC7  CD 69 F0    CALL  F069               Vorzeichen setzen
EECA  CD 7C F0    CALL  F07C               ggf. führende "*" setzen
EECD  7A          LD    A,D                Formatierungs-Flags
EECE  1F          RRA                      Formatüberlauf ?
EECF  D0          RET   NC                 nein ? dann zurück
EED0  2B          DEC   HL                   sonst "%" vor Zahl
EED1  36 25       LD    (HL),25              setzen
EED3  C9          RET
```

```
**********************************  Dezimalpunkt u. Exponenten setzen
                                    IN : C: Gesamtstellenzahl
                                         D: Formatierungsflags
                                         E: Kommaposition
                                    OUT: B: Zahl der Vorkommastellen
EED4  7A        LD    A,D              Formatierungsflags
EED5  87        ADD   A                  keine spezielle
EED6  30 29     JR    NC,EF01            Zahlenformatierung ?
EED8  FA 27 EF  JP    M,EF27           form. Exponentialdarstellung ?
EEDB  7B        LD    A,E              Kommaposition
EEDC  81        ADD   C                + Stellenzahl = dez. Exp.+1
EEDD  D6 0A     SUB   0A               Dezimalexponent <9 ?
EEDF  FA 88 EF  JP    M,EF88           dann o.k.
EEE2  16 01     LD    D,01             sonst Flag für Formatüberlauf

**********************************  normale Exponentialdarstellung
EEE4  41        LD    B,C              Gesamt.-Z. als Vorkommast.-Z.
EEE5  79        LD    A,C              Gesamtstellenzahl
EEE6  B7        OR    A                =0 ?
EEE7  28 15     JR    Z,EEFE           dann Zahl=0
EEE9  83        ADD   E                Stellenzahl +Kommaposition
EEEA  3D        DEC   A                -1 (eine Vorkommastelle)
EEEB  5F        LD    E,A              gibt Dezimalexponenten
EEEC  CD 0E F0  CALL  F00E             Nachkommanullen unterdrücken
EEEF  06 01     LD    B,01             Zahl der Vorkommastellen=1
EEF1  79        LD    A,C              neue Gesamtstellenzahl
EEF2  FE 07     CP    07               weniger als 7 Stellen ?
EEF4  38 04     JR    C,EEFA           dann o.k.
EEF6  CB 72     BIT   6,D              Flag f. max. 7 Mant.-Stellen
EEF8  20 26     JR    NZ,EF20          gesetzt ? dann nur 7 Mant.-St.
EEFA  B8        CP    B                Stellenz.=Vorkommastellenz. ?
EEFB  C4 A0 EF  CALL  NZ,EFA0          nein ? d. Dezimalpunkt setzen
EEFE  C3 62 EF  JP    EF62             Dezimal-Exponenten setzen

**********************************  normale Darstellung
EF01  7B        LD    A,E              Kommaposition (dez. Exp-8)
EF02  B7        OR    A
EF03  FA 0A EF  JP    M,EF0A           Nachkommastellen vorhanden ?
EF06  20 DC     JR    NZ,EEE4          Zahl>1E9 ? dann Exp.-Darst.
EF08  41        LD    B,C              Ges.-St.-Zahl=Vork.-St.-Zahl
EF09  C9        RET

**********************************  Zahl mit Nachkommastellen
EF0A  43        LD    B,E              Kommaposition
EF0B  CD 0E F0  CALL  F00E             Nachkomma-Nullen unterdrücken
EF0E  78        LD    A,B              neue Kommastellung
EF0F  B7        OR    A
EF10  28 F6     JR    Z,EF08           keine Nachkommastellen mehr ?
EF12  93        SUB   E                minus alten Wert
EF13  58        LD    E,B              neue Kommastellung setzen
EF14  47        LD    B,A              Zahl der abgeschn. Stellen
EF15  81        ADD   C                + Ziffernzahl=max. Ziffernzahl
EF16  83        ADD   E                + Kommastellung
EF17  FA E4 EE  JP    M,EEE4           Zahl zu klein ? d. Exp.-Darst.
EF1A  CD B4 EF  CALL  EFB4             führende Nullen in Buffer
EF1D  C3 A0 EF  JP    EFA0             Dezimalpunkt setzen
```

```
***********************************   Exp.-Darst. mit max. 7 Mant.-St.
EF20  3E 06     LD    A,06            6 Nachkommastellen
EF22  32 6E AE  LD    (AE6E),A        (1 Vorkommastelle)
EF25  18 24     JR    EF4B            Zahl formatieren

***********************************   formatierte Exponentialdarst.
EF27  06 80     LD    B,80            negatives Vorzeichen
EF29  CD 25 F0  CALL  F025            Zahl der Vorkommaziffern holen
EF2C  30 04     JR    NC,EF32         nicht zu viele Sonderzeichen ?
EF2E  CD 96 F0  CALL  F096            sonst Formatüberlauf
EF31  AF        XOR   A               keine Vorkommastelle
EF32  47        LD    B,A             Vorkommastellenzahl
EF33  CC 36 F0  CALL  Z,F036          =0 ? dann Nachkommast. holen
EF36  20 0C     JR    NZ,EF44         Nach- o. Vork.-St. gewünscht ?
EF38  04        INC   B               Vorkommastellenzahl erhöhen
EF39  3A 6E AE  LD    A,(AE6E)        gewünschte Nachk.-Stellenzahl
EF3C  B7        OR    A
EF3D  28 05     JR    Z,EF44          keine Nachkommastellen ?
EF3F  05        DEC   B               Zahl d. Vorkommastellen zurück
EF40  3C        INC   A               Nachkommastellenzahl erhöhen
EF41  32 6E AE  LD    (AE6E),A        und wieder speichern
EF44  79        LD    A,C             Gesamtstellenzahl
EF45  B7        OR    A               =0 ?
EF46  28 04     JR    Z,EF4C          dann dez. Exp.=Kommastellung
EF48  83        ADD   E               Stellenzahl+Kommastellung
EF49  90        SUB   B               -Nachkommastellenzahl
EF4A  5F        LD    E,A             gibt Dezimalexponenten
EF4B  78        LD    A,B             Vorkommastellenzahl
EF4C  F5        PUSH  AF              retten
EF4D  47        LD    B,A             Vorkommastellenzahl
EF4E  CD 8B EF  CALL  EF8B            Mantisse formatieren
EF51  F1        POP   AF              alte Vorkommastellenzahl
EF52  B8        CP    B               = neue ?
EF53  28 0D     JR    Z,EF62          d. keine zusätzl. Rundungsst.
EF55  1C        INC   E               Dezimalexponenten erhöhen
EF56  23        INC   HL              erste Ziffer übergehen
EF57  05        DEC   B               Vorkommastellenzahl erniedr.
EF58  E5        PUSH  HL              Zeiger auf Start der Zahl
EF59  7E        LD    A,(HL)          Zeichen aus Zahl
EF5A  FE 2E     CP    2E              "." ?
EF5C  20 01     JR    NZ,EF5F         nein ?
EF5E  23        INC   HL              sonst übergehen
EF5F  36 31     LD    (HL),31         "1" als 1. Ziffer (Mant./10)
EF61  E1        POP   HL              Zeiger auf Start der Zahl
EF62  3E 45     LD    A,45            "E"
EF64  CD 6F F0  CALL  F06F            ans Bufferende
EF67  7B        LD    A,E             Dezimalexponent
EF68  87        ADD   A               Vorzeichen ins Carry
EF69  3E 2B     LD    A,2B            "+"
EF6B  30 05     JR    NC,EF72         positiver Dezimalexponent ?
EF6D  AF        XOR   A                 sonst Betrag
EF6E  93        SUB   E                 des Dezimalexponenten
EF6F  5F        LD    E,A               bilden
EF70  3E 2D     LD    A,2D            "-"
EF72  CD 6F F0  CALL  F06F            Vorzeichen ans Bufferende
EF75  7B        LD    A,E             Dezimalexponent
EF76  0E 2F     LD    C,2F            "0"-1
EF78  0C        INC   C               Zehnerstelle erhöhen
```

```
EF79  D6 0A      SUB   0A          als Ausgleich 10 subtrahieren
EF7B  30 FB      JR    NC,EF78     weitere Zehnerstellen ?
EF7D  5F         LD    E,A         Einerstelle-10
EF7E  79         LD    A,C         Zehnerstelle
EF7F  CD 6F F0   CALL  F06F        ans Bufferende
EF82  7B         LD    A,E         Einerstelle-10
EF83  C6 3A      ADD   3A          +10+"0" gibt ASCII-Code
EF85  C3 6F F0   JP    F06F        ans Bufferende

*********************************  formatierte Darstellung
EF88  CD B4 EF   CALL  EFB4          führende Nullen in Buffer
EF8B  CD 36 F0   CALL  F036          gewünschte Nachkommast.-Zahl
EF8E  80         ADD   B             +Zahl der Vorkommastellen
EF8F  B9         CP    C
EF90  30 05      JR    NC,EF97       >=Gesamtstellenzahl ?
EF92  CD C8 EF   CALL  EFC8          sonst entsprechend runden
EF95  18 04      JR    EF9B
EF97  91         SUB   C             Zahl d. gew.-Z. d. exist. St.
EF98  C4 EF EF   CALL  NZ,EFEF       <>0 ? dann Nullen anhängen
EF9B  3A 6E AE   LD    A,(AE6E)      gewünschte Nachkommast.-Zahl
EF9E  B7         OR    A             =0 ?
EF9F  C8         RET   Z             dann fertig

*********************************  Dezimalpunkt einfügen
                                   IN : B: Vorkommastellenzahl
EFA0  0E 2E      LD    C,2E          "."
EFA2  78         LD    A,B           Vorkommastellenzahl
EFA3  C5         PUSH  BC            retten
EFA4  47         LD    B,A           Vorkommastellenzahl
EFA5  04         INC   B             +1=zu verschiebende St.-Zahl
EFA6  85         ADD   L
EFA7  6F         LD    L,A
EFA8  8C         ADC   H               Bufferzeiger
EFA9  95         SUB   L               addieren
EFAA  67         LD    H,A
EFAB  2B         DEC   HL            Bufferzeiger
EFAC  79         LD    A,C           voriges Zeichen
EFAD  4E         LD    C,(HL)        aktuelles Zeichen
EFAE  77         LD    (HL),A        durch voriges ersetzen
EFAF  05         DEC   B
EFB0  20 F9      JR    NZ,EFAB       weitere Stellen verschieben ?
EFB2  C1         POP   BC            Vorkommastellenzahl
EFB3  C9         RET

*********************************  führende Nullen in Buffer
                                   IN/OUT: C: Zahl der Stellen
                                           E: Kommastellung
                                   OUT: B: Vorkommastellenzahl
EFB4  7B         LD    A,E           Kommastellung
EFB5  81         ADD   C             +Gesamtstellenzahl
EFB6  47         LD    B,A           =Vorkommastellenzahl
EFB7  F0         RET   P             Vorkommastellen vorhanden ?
EFB8  2F         CPL                   Zweierkomplement, gibt Zahl
EFB9  3C         INC   A               der ben. führenden Nullen
EFBA  06 14      LD    B,14          max. Wert
EFBC  B8         CP    B             benötigte Zahl zu groß ?
EFBD  30 01      JR    NC,EFC0       dann Maximalzahl
EFBF  47         LD    B,A           sonst berechnete Zahl
```

```
EFC0  2B          DEC    HL
EFC1  36 30       LD     (HL),30       "0" in Buffer
EFC3  0C          INC    C             Gesamtstellenzahl erhöhen
EFC4  05          DEC    B
EFC5  20 F9       JR     NZ,EFC0       weitere führende Nullen ?
EFC7  C9          RET

*********************************** Zahl runden
                                    IN : A: gewünschte Stellenzahl
                                         B: Zahl der Vorkommastellen
                                         HL: Bufferzeiger
                                    OUT: B: Vorkommastellenzahl
                                         C: Gesamtstellenzahl
                                         HL: Bufferzeiger
EFC8  E5          PUSH   HL            Bufferzeiger retten
EFC9  4F          LD     C,A           gewünschte Stellenzahl
EFCA  85          ADD    L
EFCB  6F          LD     L,A             Stellenzahl zu Bufferzeiger
EFCC  8C          ADC    H               addieren, gibt Zeiger auf
EFCD  95          SUB    L               1. nicht benötigte Stelle
EFCE  67          LD     H,A
EFCF  7E          LD     A,(HL)        1. nicht benötigte Stelle
EFD0  36 00       LD     (HL),00       durch Bufferende ersetzen
EFD2  22 70 AE    LD     (AE70),HL     Zeiger auf Bufferende setzen
EFD5  FE 35       CP     35            Ziffer >="5" ?
EFD7  D4 E1 EF    CALL   NC,EFE1       dann aufrunden
EFDA  E1          POP    HL            Bufferzeiger auf Start d. Zahl
EFDB  D8          RET    C             kein Rundungsüberlauf ?
EFDC  2B          DEC    HL            sonst Stelle vor Zahl
EFDD  36 31       LD     (HL),31       auf "1" setzen
EFDF  04          INC    B             Vorkommastellenzahl erhöhen
EFE0  C9          RET

*********************************** Zahl bei letzter Stelle um 1 erh.
                                    IN : HL: Zeiger nach letzter St.
                                         C: Stellenzahl
                                    OUT: CY=1, wenn o.k.
                                         CY=0, wenn Überlauf
                                         C: Stellenzahl
EFE1  79          LD     A,C           Stellenzahl
EFE2  B7          OR     A             CY=0, keine Stelle mehr ?
EFE3  C8          RET    Z             dann Überlauf
EFE4  2B          DEC    HL            Zeiger a. nächsthöherw. Ziffer
EFE5  0D          DEC    C             Stellenzahl herunterzählen
EFE6  7E          LD     A,(HL)        Ziffer
EFE7  34          INC    (HL)          Ziffer erhöhen
EFE8  FE 39       CP     39            war Ziffer <"9" ?
EFEA  D8          RET    C             dann fertig
EFEB  36 30       LD     (HL),30       sonst Ziffer "0"
EFED  18 F2       JR     EFE1          nächsthöhere Ziffer erhöhen

*********************************** Nullen an Zahl anhängen
                                    IN : A: Zahl der Nullen
                                         HL: Bufferzeiger
                                         C: Stellenzahl
                                    OUT: HL: Bufferzeiger
EFEF  D5          PUSH   DE
EFF0  C5          PUSH   BC              Stellenzahlen retten
```

```
EFF1  EB          EX    DE,HL        Bufferzeiger nach DE
EFF2  47          LD    B,A          Zahl der Nullen
EFF3  7B          LD    A,E
EFF4  90          SUB   B            Zeiger auf Zahl
EFF5  6F          LD    L,A          - Zahl der zusätzlichen Nullen
EFF6  9F          SBC   A            gibt neuen Zeiger auf Zahl
EFF7  82          ADD   D
EFF8  67          LD    H,A
EFF9  E5          PUSH  HL           neuer Zeiger auf Zahl
EFFA  0C          INC   C            Predecrement ausgleichen
EFFB  18 04       JR    F001
EFFD  1A          LD    A,(DE)       Zeichen aus Buffer
EFFE  13          INC   DE
EFFF  77          LD    (HL),A       nach unten kopieren
F000  23          INC   HL
F001  0D          DEC   C
F002  20 F9       JR    NZ,EFFD      weitere Stellen ?
F004  36 30       LD    (HL),30      "0" hinter Zahl
F006  23          INC   HL
F007  05          DEC   B
F008  20 FA       JR    NZ,F004      weitere Nullen ?
F00A  E1          POP   HL           Zeiger auf Zahl
F00B  C1          POP   BC           und Stellenzahlen zurück
F00C  D1          POP   DE
F00D  C9          RET

*********************************** Nachkomma-Nullen unterdrücken
                                     IN/OUT: HL: Bufferzeiger
                                              C: Stellenzahl
                                              B: Kommastellung
F00E  E5          PUSH  HL           Bufferzeiger retten
F00F  2A 70 AE    LD    HL,(AE70)    Zeiger auf Ende der Zahl
F012  2B          DEC   HL
F013  7E          LD    A,(HL)       letzte Ziffer
F014  23          INC   HL
F015  FE 30       CP    30           "0" ?
F017  20 05       JR    NZ,F01E      nein ? dann fertig
F019  2B          DEC   HL           sonst Endzeiger,
F01A  0D          DEC   C            Stellenzahl
F01B  04          INC   B            und Kommastellung korrigieren
F01C  20 F4       JR    NZ,F012      weitere Nachkommastellen ?
F01E  36 00       LD    (HL),00      Bufferende neu markieren
F020  22 70 AE    LD    (AE70),HL    Zeiger auf Bufferende setzen
F023  E1          POP   HL           Bufferzeiger auf Zahl
F024  C9          RET

*********************************** Vorkommastellenzahl ohne Sonderz.
                                     OUT: A: gewünschter Wert
                                          CY=0, wenn zu viele Sonderz.
F025  CD 9B F0    CALL  F09B         Vorzeichenflags holen
F028  9F          SBC   A              Vorzeichen vor der Zahl ?
F029  3C          INC   A              dann A=1, sonst A=0
F02A  47          LD    B,A          Platz für Vorzeichen
F02B  7A          LD    A,D          Formatierungs-Flags
F02C  E6 04       AND   04           Bit für "$"
F02E  28 01       JR    Z,F031       nicht gesetzt ?
F030  04          INC   B            sonst Platz erhöhen
F031  3A 6F AE    LD    A,(AE6F)     gewünschte Vorkommastellenzahl
```

```
F034  90          SUB   B             - Platz für Sonderzeichen
F035  C9          RET

***********************************  Nachkommastellen (ohne ".")
                                     OUT: A: gewünschte Zahl
F036  3A 6E AE    LD    A,(AE6E)        gew. Nachkommastellenzahl
F039  B7          OR    A
F03A  C8          RET   Z               keine Nachkommastellen gew. ?
F03B  3D          DEC   A               sonst Stelle für "." abziehen
F03C  C9          RET

***********************************  Komma-Einteilungen setzen
                                     IN/OUT: B: Vorkommastellenzahl
                                             D: Formatierungs-Flags
                                             HL: Bufferzeiger
F03D  7A          LD    A,D             Formatierungs-Flags
F03E  E6 02       AND   02              Bit für Komma-Einteilung
F040  C8          RET   Z               nicht gesetzt ?
F041  78          LD    A,B             Zahl der Vorkommastellen
F042  D6 03       SUB   03              Position des nächsten Kommas
F044  D8          RET   C                 noch weniger als 4 Stellen ?
F045  C8          RET   Z                 dann zurück
F046  F5          PUSH  AF              Position retten
F047  0E 2C       LD    C,2C            ","
F049  CD A3 EF    CALL  EFA3            in Buffer einfügen
F04C  04          INC   B               Vorkommastellenzahl erhöhen
F04D  F1          POP   AF              Position
F04E  18 F2       JR    F042            ggf. nächstes Komma setzen

***********************************  ggf. führende Null in Buffer
                                     IN/OUT: E: Vorkommastellenzahl
                                             D: Formatierungs-Flags
                                             HL: Bufferzeiger
F050  7A          LD    A,D             Formatierungs-Flags
F051  87          ADD   A               Flag für formatierte Darst.
F052  30 07       JR    NC,F05B         nicht gesetzt ?
F054  C5          PUSH  BC
F055  CD 25 F0    CALL  F025            gew. Vork.-St.-Z. o. Sonderz.
F058  C1          POP   BC
F059  D8          RET   C               gew. Zahl <=0 ?
F05A  C8          RET   Z
F05B  3E 30       LD    A,30            sonst "0"
F05D  18 06       JR    F065            vor die Zahl in Buffer

***********************************  ggf. führendes "$" in Buffer
                                     IN/OUT: E: Vorkommastellenzahl
                                             D: Formatierungs-Flags
                                             HL: Bufferzeiger
F05F  7A          LD    A,D             Formatierungs-Flags
F060  E6 04       AND   04              Bit für "$$"
F062  C8          RET   Z               nicht gesetzt ?
F063  3E 24       LD    A,24            sonst "$"
F065  1C          INC   E               Vorkommastellenzahl erhöhen
F066  2B          DEC   HL                Zeichen vor Zahl
F067  77          LD    (HL),A            in Buffer schreiben
F068  C9          RET
```

```
**********************************   Vorzeichen setzen
                                     IN/OUT: D: Formatierungs-Flags
                                            HL: Bufferzeiger
F069  CD 9B F0      CALL    F09B        Vorzeichenflags holen
F06C  C8            RET     Z           kein Vorzeichen gewünscht ?
F06D  30 F6         JR      NC,F065     Vorzeichen vor der Zahl ?

**********************************   Zeichen ans Bufferende schreiben
                                     IN : A: Zeichen
F06F  E5            PUSH    HL
F070  2A 70 AE      LD      HL,(AE70)   Zeiger auf Bufferende
F073  77            LD      (HL),A      Zeichen nach Zahl speichern
F074  23            INC     HL
F075  36 00         LD      (HL),00     Null ans Bufferende
F077  22 70 AE      LD      (AE70),HL   Zeiger auf Bufferende setzen
F07A  E1            POP     HL
F07B  C9            RET

**********************************   führende Zeichen vor die Zahl
                                     IN/OUT: E: Vorkommastellenzahl
                                            beim CPC 664/6128:
                                              B: Vorkommastellenzahl
                                              D: Formatierungs-Flags
                                              HL: Bufferzeiger
F07C  7A            LD      A,D         Formatierungs-Flags
F07D  B7            OR      A           keine formatierte Darst. ?
F07E  F0            RET     P           dann zurück
F07F  3A 6F AE      LD      A,(AE6F)    gew. Vorkommastellenzahl
F082  93            SUB     E           - tatsächliche Zahl
F083  C8            RET     Z           gleich ? dann fertig
F084  38 10         JR      C,F096      zuviele St. ? d. Formatüberl.
F086  47            LD      B,A         Zahl der zusätzlichen Stellen
F087  7A            LD      A,D         Formatierungs-Flags
F088  E6 20         AND     20          Bit für "**"
F08A  3E 2A         LD      A,2A        "*"
F08C  20 02         JR      NZ,F090     Flag für "**" gesetzt ?
F08E  3E 20         LD      A,20        sonst Space
F090  2B            DEC     HL            Zeichen vor die Zahl
F091  77            LD      (HL),A        in Buffer schreiben
F092  05            DEC     B
F093  20 FB         JR      NZ,F090     weitere führende Zeichen ?
F095  C9            RET

**********************************   Flag für Formatüberlauf setzen
F096  7A            LD      A,D
F097  F6 01         OR      01
F099  57            LD      D,A
F09A  C9            RET

**********************************   Vorzeichenflags holen
                                     IN : D: Formatierungsflags
                                          B: Vorzeichen
                                     OUT: A: ASCII-Code d. Vorzeichens
                                          Z=1, wenn kein Vorzeichen
                                          CY=1, w. Vorz. nach der Zahl
F09B  78            LD      A,B         Vorzeichen
F09C  06 2D         LD      B,2D        "-"
F09E  87            ADD     A           Vorzeichen ins Carry
```

```
F09F  38 0F       JR    C,F0B0     negativ ?
F0A1  7A          LD    A,D        Formatierungsflags
F0A2  E6 98       AND   98           Vorz. vor d. Zahl, nicht "+"
F0A4  EE 80       XOR   80           und formatierte Darst. ?
F0A6  37          SCF
F0A7  C8          RET   Z          dann kein Vorzeichen
F0A8  06 2B       LD    B,2B       "+"
F0AA  E6 08       AND   08         Flag für "+" bei pos. Vorz.
F0AC  20 02       JR    NZ,F0B0    gesetzt ?
F0AE  06 20       LD    B,20       sonst Space
F0B0  7A          LD    A,D        Formatierungsflags
F0B1  F6 EF       OR    EF           Flag für Vorz. nach der Zahl
F0B3  C6 10       ADD   10           ins Carry
F0B5  78          LD    A,B        ASCII-Code des Vorzeichens
F0B6  C9          RET

*********************************** Binärzahl nach ASCII-Mantisse
                                   IN : DE: Zeiger auf höchstw. Byte
                                            der Binärzahl
                                        C: Länge der Binärzahl
                                        HL: Zeiger auf ASCII-Buffer
                                   OUT: HL: Bufferzeiger
                                        C: Zahl der Ziffern
F0B7  E5          PUSH  HL         Bufferzeiger retten
F0B8  EB          EX    DE,HL      Zeiger auf h. Zahlbyte nach HL
F0B9  CD DD F0    CALL  F0DD       Zahl nach BCD wandeln
F0BC  E1          POP   HL         Bufferzeiger zurück
F0BD  78          LD    A,B        Länge der BCD-Zahl
F0BE  87          ADD   A          mal 2, da 2 Ziffern pro Byte
F0BF  4F          LD    C,A        gibt Zahl der Ziffern
F0C0  C8          RET   Z          keine Ziffern ? dann fertig
F0C1  1A          LD    A,(DE)     Byte aus BCD-Zahl
F0C2  E6 0F       AND   0F         unteres Nibble
F0C4  C6 30       ADD   30         nach ASCII
F0C6  2B          DEC   HL
F0C7  77          LD    (HL),A     in Buffer schreiben
F0C8  1A          LD    A,(DE)     Byte aus BCD-Zahl
F0C9  E6 F0       AND   F0         oberes Nibble
F0CB  1F          RRA
F0CC  1F          RRA                oberes Nibble
F0CD  1F          RRA                in unteres Nibble
F0CE  1F          RRA                schieben
F0CF  C6 30       ADD   30         ASCII-Code herstellen
F0D1  2B          DEC   HL
F0D2  77          LD    (HL),A     in Buffer schreiben
F0D3  13          INC   DE         BCD-Bufferzeiger
F0D4  05          DEC   B
F0D5  20 EA       JR    NZ,F0C1    weitere BCD-Bytes ?
F0D7  FE 30       CP    30         führende Ziffer ="0" ?
F0D9  C0          RET   NZ         nein ? dann zurück
F0DA  0D          DEC   C          sonst Null unterdrücken
F0DB  23          INC   HL         Zeiger auf nächste Zahl
F0DC  C9          RET

*********************************** Binärzahl nach BCD wandeln
                                   IN : HL: Zeiger auf höchstw. Byte
                                            der Binärzahl
                                        C: Länge der Binärzahl
```

```
                                        OUT: DE: Zeiger auf BCD-Zahl
                                                 B: Länge der BCD-Zahl
F0DD  11 46 AE    LD    DE,AE46         Zeiger auf BCD-Buffer
F0E0  AF          XOR   A               Null
F0E1  47          LD    B,A             als Zahl der BCD-Stellen
F0E2  B6          OR    (HL)            Byte aus Binärzahl
F0E3  2B          DEC   HL              Zg. nächstniederwertiges Byte
F0E4  20 04       JR    NZ,F0EA         Byte signifikant ?
F0E6  0D          DEC   C               Zähler f. signifikante Bytes
F0E7  20 F9       JR    NZ,F0E2         weitere Bytes ?
F0E9  C9          RET
F0EA  37          SCF                   Byteende-Kennzeichen
F0EB  8F          ADC   A                 nächstes gesetztes
F0EC  30 FD       JR    NC,F0EB           Bit suchen
F0EE  EB          EX    DE,HL           BCD-Zg. n. HL, Bin.-Zg. n. DE
F0EF  D5          PUSH  DE              Binärzahl-Zeiger retten
F0F0  57          LD    D,A             Byte aus Binärzahl
F0F1  18 11       JR    F104            auswerten
F0F3  1A          LD    A,(DE)          nächstes Binärzahl-Byte
F0F4  1B          DEC   DE                Zg. a. nächstniederwertiges
F0F5  D5          PUSH  DE                Byte retten
F0F6  37          SCF                   Byteende-Kennzeichen
F0F7  8F          ADC   A               nächstes Bit aus Binärzahl
F0F8  57          LD    D,A             Binärzahl-Byte
F0F9  58          LD    E,B             Zahl der BCD-Bytes
F0FA  7E          LD    A,(HL)          BCD-Byte
F0FB  8F          ADC   A               Bit a. Zahl/Übertr. hineinrot.
F0FC  27          DAA                   wieder nach BCD
F0FD  77          LD    (HL),A          BCD-Byte wieder speichern
F0FE  23          INC   HL              Zeiger auf nächstes BCD-Byte
F0FF  1D          DEC   E
F100  20 F8       JR    NZ,F0FA         weitere BCD-Bytes ?
F102  30 03       JR    NC,F107         Übertrag zu nächst. BCD-Byte ?
F104  04          INC   B               Länge der BCD-Zahl erhöhen
F105  36 01       LD    (HL),01         Übertrag setzen
F107  21 46 AE    LD    HL,AE46         Zeiger auf BCD-Zahl
F10A  7A          LD    A,D             Byte aus Binär-Zahl
F10B  87          ADD   A               nächstes Bit holen
F10C  20 EA       JR    NZ,F0F8         kein Byteende ? dann auswerten
F10E  D1          POP   DE              Zeiger in Binärzahl
F10F  0D          DEC   C               Zähler für Binärzahl-Bytes
F110  20 E1       JR    NZ,F0F3         Binärzahl noch nicht zu Ende ?
F112  EB          EX    DE,HL           Zeiger auf BCD-Zahl nach DE
F113  C9          RET

***********************************     Zahl nach Binärstring wandeln
                                        IN : HL: Zeiger auf Zahl
                                                 C: Typ der Zahl
                                                 B: min. Stellenzahl
                                        OUT: DE: Zeiger auf String
                                                 HL: Zeiger nach Zahl
F114  11 01 01    LD    DE,0101           Bits/Stelle, Bitmaske f. 1 St.
F117  18 03       JR    F11C

***********************************     Zahl nach Hex-String wandeln
                                        IN : HL: Zeiger auf Zahl
                                                 C: Typ der Zahl
                                                 B:: min. Stellenzahl
```

```
                                   OUT: DE: Zeiger auf String
                                           HL: Zeiger nach Zahl
F119  11 0F 04    LD    DE,040F      Bits/Stelle, Bitmaske f. 1 St.
F11C  D5          PUSH  DE           retten
F11D  79          LD    A,C          Typ der Zahl
F11E  CD 4B FF    CALL  FF4B         Zahl nach FAC kopieren
F121  E3          EX    (SP),HL      Zg. nach Zahl r., Param. zur.
F122  E5          PUSH  HL           Bits/Stelle, Bitmaske f. 1 St.
F123  C5          PUSH  BC           min. Stellenzahl
F124  CD C2 FE    CALL  FEC2         UNT, FAC nach Integer nach HL
F127  11 57 AE    LD    DE,AE57      Zeiger auf Buffer für String
F12A  AF          XOR   A            Null
F12B  12          LD    (DE),A       als Endekennzeichen in Buffer
F12C  F1          POP   AF           min. Stellenzahl
F12D  C1          POP   BC           Bits/Stelle, Bitmaske f. 1 St.
F12E  D6 01       SUB   01           Stellenzähler
F130  CE 00       ADC   00           unter 0 ? dann wieder auf 0
F132  F5          PUSH  AF           Stellenzähler retten
F133  7D          LD    A,L          Lo-Byte der Zahl
F134  A1          AND   C            Bit(s) für diese Stelle isol.
F135  F6 F0       OR    F0             $30 addieren,
F137  27          DAA                  wenn Ziffer <=9,
F138  C6 A0       ADD   A0             sonst $37 addieren,
F13A  CE 40       ADC   40             gibt ASCII-Code
F13C  1B          DEC   DE           Bufferzeiger für String
F13D  12          LD    (DE),A       Ziffer an Bufferanfang setzen
F13E  7D          LD    A,L          Lo-Byte der Zahl
F13F  B1          OR    C              Bit(s) für diese Stelle
F140  A9          XOR   C              löschen
F141  6F          LD    L,A          Lo-Byte wieder zurück
F142  B4          OR    H            Zahl=0 ?
F143  28 0E       JR    Z,F153       dann Test auf Ende
F145  C5          PUSH  BC           Bits/Stelle, Maske f. 1 Stelle
F146  7C          LD    A,H
F147  1F          RRA
F148  67          LD    H,A            Zahl nach rechts
F149  7D          LD    A,L            rotieren
F14A  1F          RRA
F14B  6F          LD    L,A
F14C  05          DEC   B            weitere Bits in dies. Stelle ?
F14D  20 F7       JR    NZ,F146
F14F  C1          POP   BC           Bits/Stelle, Maske f. 1 Stelle
F150  F1          POP   AF           Stellenzähler
F151  18 DB       JR    F12E         nächste Stelle wandeln
F153  F1          POP   AF           Stellenzähler
F154  20 D8       JR    NZ,F12E      weitere Stellen gewünscht ?
F156  E1          POP   HL           Zeiger nach eingegebener Zahl
F157  C9          RET

*********************************** Basic-Funktion PEEK
F158  CD C2 FE    CALL  FEC2         UNT, FAC nach Integer nach HL
F15B  E7          RST   20           Byte aus RAM laden
F15C  C3 0A FF    JP    FF0A         und in FAC speichern

*********************************** Basic-Befehl POKE
F15F  CD 91 CE    CALL  CE91         Adresse nach DE holen
F162  D5          PUSH  DE           und retten
F163  CD 37 DD    CALL  DD37         Test auf Komma
```

```
F166  2C                                  ","
F167  CD 67 CE    CALL   CE67             Byte nach A holen
F16A  D1          POP    DE               Adresse
F16B  12          LD     (DE),A           Byte speichern
F16C  C9          RET

*********************************** Basic-Funktion INP
F16D  CD 8D FE    CALL   FE8D             CINT, FAC nach Integer nach HL
F170  44          LD     B,H                nach BC
F171  4D          LD     C,L
F172  ED 78       IN     A,(C)            Byte laden
F174  C3 0A FF    JP     FF0A             und in FAC speichern

*********************************** Basic-Befehl OUT
F177  CD 94 F1    CALL   F194             Parameter holen
F17A  ED 79       OUT    (C),A            und Byte ausgeben
F17C  C9          RET

*********************************** Basic-Befehl WAIT
F17D  CD 94 F1    CALL   F194             Parameter holen
F180  57          LD     D,A              Byte nach D
F181  1E 00       LD     E,00             Default für 3. Parameter
F183  28 08       JR     Z,F18D           Statementende ? dann Default
F185  CD 37 DD    CALL   DD37             Test auf Komma
F188  2C                                  ","
F189  CD 67 CE    CALL   CE67             Byte nach A holen
F18C  5F          LD     E,A
F18D  ED 78       IN     A,(C)            Byte laden
F18F  AB          XOR    E                  mit Werten
F190  A2          AND    D                  verknüpfen
F191  28 FA       JR     Z,F18D           ggf. weiter warten
F193  C9          RET

*********************************** Adresse und Byte holen
                                     OUT: BC: Adresse; A: Byte
F194  CD 91 CE    CALL   CE91             Adresse nach DE
F197  42          LD     B,D                nach BC
F198  4B          LD     C,E
F199  CD 37 DD    CALL   DD37             Test auf Komma
F19C  2C                                  ","
F19D  C3 67 CE    JP     CE67             Byte nach A holen

*********************************** RSX-Wort auswerten
F1A0  23          INC    HL               RSX-Code übergehen
F1A1  7E          LD     A,(HL)           folgendes Zeichen
F1A2  B7          OR     A                unbekannter Code ?
F1A3  20 10       JR     NZ,F1B5          dann Fehler
F1A5  23          INC    HL               Code übergehen
F1A6  E5          PUSH   HL               Zeiger auf Befehlsnamen retten
F1A7  CD D4 BC    CALL   BCD4             KL FIND COMMAND
F1AA  EB          EX     DE,HL            Routinenadresse nach DE
F1AB  E1          POP    HL               Zeiger auf Befehlsnamen
F1AC  30 07       JR     NC,F1B5          Befehl nicht gefunden ?
F1AE  7E          LD     A,(HL)           Byte aus Namen
F1AF  23          INC    HL
F1B0  17          RLA                     Ende ?
F1B1  30 FB       JR     NC,F1AE          sonst weiter
F1B3  18 0A       JR     F1BF             Parameter holen, ausführen
```

```
F1B5  1E 1C        LD     E,1C          Nr. für "Unknown command"
F1B7  C3 94 CA     JP     CA94          Fehler ausgeben

********************************** Basic-Befehl CALL
F1BA  CD 91 CE     CALL   CE91          Adresse holen
F1BD  0E FF        LD     C,FF          ROM-Konfig., ROMs aus

********************************** Parameter holen, Routine ausf.
                                     IN : DE: Routinenadresse; C: ROM-Konfiguration
F1BF  ED 53 72 AE  LD     (AE72),DE     Adresse speichern
F1C3  79           LD     A,C           ROM-Konfiguration
F1C4  32 74 AE     LD     (AE74),A      speichern
F1C7  ED 73 77 AE  LD     (AE77),SP     Stackpointer retten
F1CB  06 20        LD     B,20          max. Parameteranzahl
F1CD  CD 55 DD     CALL   DD55          folgt Komma ?
F1D0  30 06        JR     NC,F1D8       nein ? dann Ende der Parameter
F1D2  CD 91 CE     CALL   CE91          Integerwert holen
F1D5  D5           PUSH   DE            auf Stack
F1D6  10 F5        DJNZ   F1CD          weitere Parameter möglich ?
F1D8  CD 4A DD     CALL   DD4A          Test auf Statementende
F1DB  22 75 AE     LD     (AE75),HL     Basic-PC retten
F1DE  3E 20        LD     A,20          max. Parameteranzahl
F1E0  90           SUB    B             -Rest = Zahl der Parameter
F1E1  DD 21 00 00  LD     IX,0000         Zeiger auf Parameter
F1E5  DD 39        ADD    IX,SP           nach IX
F1E7  DF           RST    18            Routine ausführen
F1E8  72 AE                             AE72, Zeiger auf Adr./Konfig.
F1EA  ED 7B 77 AE  LD     SP,(AE77)     Stackpointer wieder zurück
F1EE  2A 75 AE     LD     HL,(AE75)     Basic-PC wieder zurück
F1F1  C9           RET

********************************** ZONE-Default setzen
F1F2  3E 0D        LD     A,0D          13
F1F4  18 03        JR     F1F9          ZONE 13

********************************** Basic-Befehl ZONE
F1F6  CD 6D CE     CALL   CE6D          Byte holen
F1F9  32 79 AE     LD     (AE79),A      als Tabulatorweite setzen
F1FC  C9           RET

********************************** Basic-Befehl PRINT
F1FD  CD C6 C1     CALL   C1C6          opt. Streamnr. holen/setzen
F200  F5           PUSH   AF            alte Nr. retten
F201  CD 08 F2     CALL   F208          PRINT-Befehl
F204  F1           POP    AF            alte Streamnr.
F205  C3 A2 C1     JP     C1A2          wieder setzen

********************************** PRINT Fortsetzung
F208  CD 51 DD     CALL   DD51          Statementende ?
F20B  DA 4E C3     JP     C,C34E        dann Linefeed ausgeben
F20E  FE ED        CP     ED            Token für USING ?
F210  CA C4 F2     JP     Z,F2C4        dann PRINT USING
F213  EB           EX     DE,HL         Baisc-PC nach DE
F214  21 24 F2     LD     HL,F224       Zeiger auf Tabelle
F217  CD 93 FF     CALL   FF93          Adresse entspr. Token holen
F21A  EB           EX     DE,HL         Adr. nach DE, PC nach HL
F21B  CD FB FF     CALL   FFFB          Routine ausführen
F21E  CD 51 DD     CALL   DD51          Statementende ?
```

```
F221  30 EB       JR    NC,F20E      nein ? dann weiter
F223  C9          RET

*********************************** Tabelle für PRINT
F224  05                                Länge der Tabelle
F225  33 F2                             Default-Adresse
F227  2C                                ","
F228  5C F2                             Komma-Tabulator
F22A  E5                                Token für SPC
F22B  77 F2                             Routine für SPC
F22D  EA                                Token für TAB
F22E  80 F2                             Routine für TAB
F230  3B                                ";"
F231  3F DD                             nächstes Zeichen holen

*********************************** PRINT, Ausdruck ausgeben
F233  CD FB CE    CALL  CEFB            Ausdruck holen
F236  F5          PUSH  AF              Flag für Statementende
F237  E5          PUSH  HL              und Basic-PC retten
F238  CD 45 FF    CALL  FF45            String ?
F23B  28 0C       JR    Z,F249          dann ausgeben
F23D  CD 9D EE    CALL  EE9D            FAC nach ASCII wandeln
F240  CD DC F7    CALL  F7DC            String auf Stringstack
F243  36 20       LD    (HL),20         String mit Space abschließen
F245  2A C2 B0    LD    HL,(B0C2)       Zeiger auf Descriptor
F248  34          INC   (HL)            Länge erhöhen
F249  2A C2 B0    LD    HL,(B0C2)       Zeiger auf Descriptor
F24C  7E          LD    A,(HL)          Stringlänge
F24D  CD B9 C2    CALL  C2B9            paßt String noch in Zeile ?
F250  D4 4E C3    CALL  NC,C34E         sonst Linefeed ausgeben
F253  CD 28 F8    CALL  F828            String vom Stack, ausgeben
F256  E1          POP   HL              Basic-PC
F257  F1          POP   AF              Flag für Statementende
F258  CC 4E C3    CALL  Z,C34E          Statementende ? d. LF ausgeben
F25B  C9          RET

*********************************** PRINT, Komma-Tabulator
F25C  CD 3F DD    CALL  DD3F            Komma übergehen
F25F  3A 79 AE    LD    A,(AE79)        Tabulatorweite
F262  4F          LD    C,A             nach C
F263  CD 90 C2    CALL  C290            akt. Ausgabeposition holen
F266  3D          DEC   A                 Zahl der Zeichen bis zur
F267  91          SUB   C                 nächsten Tabulatorposition
F268  30 FD       JR    NC,F267           berechnen
F26A  2F          CPL                       Zweier-
F26B  3C          INC   A                   komplement
F26C  47          LD    B,A             Zahl der Zeichen
F26D  81          ADD   C               Tabulatorweite addieren
F26E  CD B9 C2    CALL  C2B9            paßt dies noch in Zeile ?
F271  D2 4E C3    JP    NC,C34E         nein ? dann Linefeed ausgeben
F274  78          LD    A,B               sonst entsprechend viele
F275  18 1E       JR    F295              Spaces ausgeben

*********************************** PRINT SPC
F277  CD A0 F2    CALL  F2A0            Integer in Klammern nach DE
F27A  CD AF F2    CALL  F2AF            Wert MOD Ausgabebreite
F27D  7B          LD    A,E             Lo-Byte
F27E  18 15       JR    F295            entsprechend viele Spaces
```

```
**********************************    PRINT TAB
F280  CD A0 F2    CALL  F2A0           Integer in Klammern nach DE
F283  1B          DEC   DE             -1 (in absolute Koordinaten)
F284  CD AF F2    CALL  F2AF           Wert MOD Ausgabebreite
F287  CD 90 C2    CALL  C290           akt. Ausgabeposition holen
F28A  2F          CPL                    Zweierkomplement
F28B  3C          INC   A              bilden. relative Koordinaten
F28D  83          ADD   E              Wert - akt. Position
F28E  38 05       JR    C,F295         positiv ? dann Spaces ausgeben
F290  CD 4E C3    CALL  C34E           sonst Linefeed ausgeben
F293  1D          DEC   E              Wert in absoluten Koordinaten
F294  7B          LD    A,E            entspr. viele Spaces ausgeben

**********************************    Spaces ausgeben
                                      IN : A: Zahl der Spaces
F295  47          LD    B,A            Zahl der Spaces
F296  04          INC   B              zum Ausgleich erhöhen
F297  05          DEC   B              Zähler
F298  C8          RET   Z              keine weiteren Spaces ?
F299  3E 20       LD    A,20           Spaces
F29B  CD 56 C3    CALL  C356           ausgeben
F29E  18 F7       JR    F297           weiter ausgeben

**********************************    Integer in Klammern holen
F2A0  CD 3F DD    CALL  DD3F           nächstes Zeichen
F2A3  CD 37 DD    CALL  DD37           Test auf Klammer auf
F2A6  28                               "("
F2A7  CD 86 CE    CALL  CE86           Integerwert holen
F2AA  CD 37 DD    CALL  DD37           Test auf Klammer zu
F2AD  29                               ")"
F2AE  C9          RET

**********************************    Zahl der Spaces MOD Ausgabebreite
                                      IN : DE: Anzahl der Spaces
                                      OUT: DE: neuer Wert
F2AF  7A          LD    A,D
F2B0  17          RLA                  Wert positiv ?
F2B1  30 03       JR    NC,F2B6        dann o.k.
F2B3  11 00 00    LD    DE,0000        sonst durch Null ersetzen
F2B6  CD 9F C2    CALL  C29F           akt. Ausgabebreite holen
F2B9  D0          RET   NC             keine gültige Breite ?
F2BA  E5          PUSH  HL             Basic-PC
F2BB  EB          EX    DE,HL          Zahl der Spaces nach HL
F2BC  5F          LD    E,A            Breite lo
F2BD  16 00       LD    D,00           Breite hi=0
F2BF  CD C1 BD    CALL  BDC1           teilen, Rest nach DE
F2C2  E1          POP   HL             Basic-PC
F2C3  C9          RET

**********************************    PRINT USING
F2C4  CD 3F DD    CALL  DD3F           USING übergehen
F2C7  CD A5 CE    CALL  CEA5           Formatstring holen
F2CA  CD 37 DD    CALL  DD37           Test auf ";"
F2CD  3B                               ";"
F2CE  E5          PUSH  HL             Basic-PC retten
F2CF  2A C2 B0    LD    HL,(B0C2)      Zeiger auf Descriptor
F2D2  7E          LD    A,(HL)         Stringlänge
```

```
F2D3  B7        OR    A          Länge =0 ?
F2D4  28 75     JR    Z,F34B     dann "Improper argument"
F2D6  E3        EX    (SP),HL    Descr.-Zg. retten, PC zurück
F2D7  CD FB CE  CALL  CEFB       Ausdruck holen
F2DA  AF        XOR   A            Flag für Ausgabe
F2DB  32 7A AE  LD    (AE7A),A     setzen
F2DE  D1        POP   DE         Descriptorzeiger
F2DF  D5        PUSH  DE
F2E0  EB        EX    DE,HL      nach HL, PC nach DE
F2E1  46        LD    B,(HL)     Stringlänge nach B
F2E2  23        INC   HL
F2E3  7E        LD    A,(HL)       Adresse
F2E4  23        INC   HL           nach HL
F2E5  66        LD    H,(HL)
F2E6  6F        LD    L,A
F2E7  EB        EX    DE,HL      Adr. nach DE, PC nach HL
F2E8  CD 24 F3  CALL  F324       Ausdruck formatiert ausgeben
F2EB  30 5E     JR    NC,F34B    Fehler? d. "Improper argument"
F2ED  CD 51 DD  CALL  DD51       Statementende ?
F2F0  38 1D     JR    C,F30F     dann Ausdruck und LF ausgeben
F2F2  FE 3B     CP    3B         ";" ?
F2F4  28 04     JR    Z,F2FA
F2F6  FE 2C     CP    2C         "," ?
F2F8  20 4C     JR    NZ,F346    nein ? dann "Syntax error"
F2FA  CD 3F DD  CALL  DD3F       nächstes Zeichen
F2FD  28 10     JR    Z,F30F     Statementende ?
F2FF  D5        PUSH  DE
F300  CD FB CE  CALL  CEFB       Ausdruck holen
F303  D1        POP   DE
F304  78        LD    A,B        restliche Stringlänge
F305  B7        OR    A          Formatstring zu Ende ?
F306  28 D6     JR    Z,F2DE     dann neu durchgehen
F308  CD 24 F3  CALL  F324       Ausdurck formatiert ausgeben
F30B  30 D1     JR    NC,F2DE    kein Ausdruck ausgegeben ?
F30D  18 DE     JR    F2ED       sonst nächsten Ausdruck ausg.
F30F  F5        PUSH  AF         Flag für "," bzw. ";" retten
F310  3E FF     LD    A,FF         Flag für keine Ausgabe
F312  32 7A AE  LD    (AE7A),A     setzen
F315  78        LD    A,B        restliche Stringlänge
F316  B7        OR    A          Formatstring nicht zu Ende ?
F317  C4 24 F3  CALL  NZ,F324    dann Ausdruck ausgeben
F31A  F1        POP   AF         Flag für "," bzw. ";"
F31B  DC 4E C3  CALL  C,C34E     nicht "," oder ";" ? dann LF
F31E  E3        EX    (SP),HL    Descriptor-Zeiger zur., PC r.
F31F  CD E8 FB  CALL  FBE8       Formatstring vom Stringstack
F322  E1        POP   HL         Basic-PC zurück
F323  C9        RET

*********************************** Ausdruck formatiert ausgeben
                                 IN : DE: Formatstringadresse
                                       B: restl. Formatstringlänge
                                 OUT: CY=0 bei Formatstringende
F324  E5        PUSH  HL           Basic-PC retten
F325  1A        LD    A,(DE)       Zeichen aus Formatstring
F326  FE 5F     CP    5F           Underscore ?
F328  20 09     JR    NZ,F333      nein ?
F32A  78        LD    A,B          restl. Stringlänge
F32B  FE 02     CP    02           <2 ?
```

```
F32D  38 0C      JR    C,F33B      dann Underscore ausgeben
F32F  13         INC   DE            nächstes
F330  05         DEC   B             Zeichen
F331  18 08      JR    F33B        ausgeben
F333  CD 50 F3   CALL  F350        ggf. String auswerten/ausgeben
F336  D4 A3 F3   CALL  NC,F3A3     kein String ? dann num. Ausdr.
F339  38 09      JR    C,F344      Ausdruck ausgegeben ?
F33B  1A         LD    A,(DE)      sonst Zeichen aus Formatstring
F33C  CD 56 C3   CALL  C356        so ausgeben
F33F  13         INC   DE            nächstes
F340  05         DEC   B             Zeichen
F341  20 E2      JR    NZ,F325     weitere Zeichen ?
F343  B7         OR    A           CY=0 für Formatstringende
F344  E1         POP   HL          Basic-PC zurück
F345  C9         RET

F346  1E 02      LD    E,02        Nr. für "Syntax error"
F348  C3 94 CA   JP    CA94        Fehler ausgeben

F34B  1E 05      LD    E,05        Nr. für "Improper argument"
F34D  C3 94 CA   JP    CA94        Fehler ausgeben

***********************************        ggf. String ausgeben
                                   IN : DE: Formatstringadresse
                                         B: restl. Formatstringlänge
                                   OUT: CY=0 für kein String
F350  1A         LD    A,(DE)        Zeichen aus Formatstring
F351  FE 21      CP    21            "!" ?
F353  0E 01      LD    C,01          Zähler für 1 Zeichen
F355  28 21      JR    Z,F378        "!" ? dann 1. Zeichen ausgeben
F357  FE 26      CP    26            "&" ?
F359  0E 00      LD    C,00          Flag für Gesamtstring
F35B  28 1B      JR    Z,F378        "&" ? dann String ausgeben
F35D  EE 5C      XOR   5C            "\" ? (CY=0!)
F35F  C0         RET   NZ            nein ? dann kein Stringformat
F360  C5         PUSH  BC
F361  D5         PUSH  DE
F362  0E 02      LD    C,02          2 Zeichen (Zweimal "\")
F364  13         INC   DE              nächstes
F365  05         DEC   B               Zeichen
F366  28 0A      JR    Z,F372        Formatstring zu Ende ?
F368  1A         LD    A,(DE)        Zeichen aus Formatstring
F369  FE 5C      CP    5C            "\" ?
F36B  28 09      JR    Z,F376        dann String ausgeben
F36D  0C         INC   C             Zähler erhöhen
F36E  FE 20      CP    20            Space ?
F370  28 F2      JR    Z,F364        dann weiter prüfen
F372  D1         POP   DE              sonst Zeiger wieder
F373  C1         POP   BC              auf 1. "\"
F374  B7         OR    A             CY=1 für kein Stringformat
F375  C9         RET
F376  F1         POP   AF              Stringparameter
F377  F1         POP   AF              löschen

***********************************        String für USING ausgeben
                                   IN : C: auszugebende Länge
F378  13         INC   DE              nächstes
F379  05         DEC   B               Zeichen
```

```
F37A  C5        PUSH  BC          Formatstringparameter
F37B  D5        PUSH  DE          retten
F37C  3A 7A AE  LD    A,(AE7A)   Flag für Ausgabe
F37F  B7        OR    A
F380  20 1D     JR    NZ,F39F    nicht ausgeben ?
F382  CD 3C FF  CALL  FF3C       Test auf Stringausdruck
F385  79        LD    A,C        Zahl der auszugebenden Zeichen
F386  B7        OR    A          =0, wenn Gesamtstring
F387  F5        PUSH  AF         Flag für Gesamtstring retten
F388  41        LD    B,C        Zahl der Zeichen nach B
F389  0E 00     LD    C,00       Startposition f. Teilstring =0
F38B  2A C2 B0  LD    HL,(B0C2)  Zeiger auf Descriptor
F38E  EB        EX    DE,HL      nach DE
F38F  C4 71 F9  CALL  NZ,F971    ggf. Teilstring holen
F392  CD 28 F8  CALL  F828       String ausgeben, vom Stack
F395  F1        POP   AF         Flag für Gesamtstring
F396  28 07     JR    Z,F39F     Gesamtstring ausgegeben ?
F398  2A C2 B0  LD    HL,(B0C2)  Zeiger auf Descriptor
F39B  96        SUB   (HL)       gew. - tatsächliche Länge
F39C  CD 95 F2  CALL  F295       entspr. viele Spaces ausgeben
F39F  D1        POP   DE          Formatstring-Parameter
F3A0  C1        POP   BC          zurück
F3A1  37        SCF              CY=1 für String ausgegeben
F3A2  C9        RET

*********************************   numerischen Ausdruck ausgeben
                                    IN : DE: Formatstringadresse
                                          B: restl. Formatstringlänge
                                    OUT: CY=0 für Fehler
F3A3  CD BA F3  CALL  F3BA          Formatstring auswerten
F3A6  D0        RET   NC            Fehler ?
F3A7  3A 7A AE  LD    A,(AE7A)      Flag für Ausgabe
F3AA  B7        OR    A             keine Ausgabe ?
F3AB  20 0B     JR    NZ,F3B8       dann fertig
F3AD  C5        PUSH  BC
F3AE  D5        PUSH  DE
F3AF  79        LD    A,C           Formatierungsflags
F3B0  CD 9F EE  CALL  EE9F          Zahl formatiert nach ASCII
F3B3  CD 41 C3  CALL  C341          und ausgeben
F3B6  D1        POP   DE
F3B7  C1        POP   BC
F3B8  37        SCF                 CY=1 für kein Fehler
F3B9  C9        RET

*********************************   Formatstr. f. numer. Ausdr. ausw.
                                    IN/OUT: DE: Formatstringadresse
                                             B: restliche Länge
                                    OUT: C: Formatierungsflags
                                         H: gew. Vorkommastellenzahl
                                         L: gew. Nachkommastellenzahl
                                         CY=0 für Fehler
F3BA  C5        PUSH  BC             Formatstring-Parameter
F3BB  D5        PUSH  DE             retten
F3BC  0E 80     LD    C,80          Flag für formatierte Ausgabe
F3BE  26 00     LD    H,00          Vorkommastellenzahl=0
F3C0  1A        LD    A,(DE)        Zeichen aus Formatstring
F3C1  FE 2B     CP    2B            "+" ?
F3C3  20 07     JR    NZ,F3CC       nein ?
```

```
F3C5  13          INC   DE           nächstes
F3C6  05          DEC   B            Zeichen
F3C7  28 23       JR    Z,F3EC     Formatstringende ? dann Fehler
F3C9  24          INC   H          Vorkommastellenz. für "+" erh.
F3CA  0E 88       LD    C,88       Flag für "+" bei pos. Vorz.
F3CC  1A          LD    A,(DE)     Zeichen aus Formatstring
F3CD  FE 2E       CP    2E         "." ?
F3CF  28 1F       JR    Z,F3F0
F3D1  FE 23       CP    23         "#" ?
F3D3  28 39       JR    Z,F40E
F3D5  13          INC   DE           nächstes
F3D6  05          DEC   B            Zeichen
F3D7  28 13       JR    Z,F3EC     Formatstringende ? dann Fehler
F3D9  EB          EX    DE,HL
F3DA  BE          CP    (HL)       Zeichen mit nächstem vergl.
F3DB  EB          EX    DE,HL
F3DC  20 0E       JR    NZ,F3EC    ungleich ? dann Fehler
F3DE  24          INC   H            Vorkommastellenzahl
F3DF  24          INC   H            um zwei erhöhen
F3E0  2E 04       LD    L,04       Flag für "$$"
F3E2  FE 24       CP    24         "$" ?
F3E4  28 23       JR    Z,F409     dann auswerten
F3E6  2E 20       LD    L,20       Flag für "**"
F3E8  FE 2A       CP    2A         "*" ?
F3EA  28 11       JR    Z,F3FD     dann auswerten
F3EC  D1          POP   DE           Formatstringparameter
F3ED  C1          POP   BC           wieder zurück
F3EE  B7          OR    A          CY=0 für Fehler
F3EF  C9          RET

F3F0  13          INC   DE           nächstes
F3F1  05          DEC   B            Zeichen
F3F2  28 F8       JR    Z,F3EC     Formatstringende ? dann Fehler
F3F4  1A          LD    A,(DE)     Zeichen aus Formatstring
F3F5  FE 23       CP    23         "#" ?
F3F7  20 F3       JR    NZ,F3EC    nein ? dann Fehler
F3F9  1B          DEC   DE           Zeichen
F3FA  04          INC   B            davor
F3FB  18 11       JR    F40E       weiter auswerten
F3FD  13          INC   DE           nächstes
F3FE  05          DEC   B            Zeichen
F3FF  28 0A       JR    Z,F40B     Formatstringende ?
F401  1A          LD    A,(DE)     nächstes Zeichen
F402  FE 24       CP    24         "$" ?
F404  20 05       JR    NZ,F40B    nicht "**$" ?
F406  24          INC   H          Vorkommastellenzahl erhöhen
F407  2E 24       LD    L,24       Flags für "**" und "$$"
F409  13          INC   DE           nächstes
F40A  05          DEC   B            Zeichen
F40B  79          LD    A,C        Formatierungsflags
F40C  B5          OR    L          entspr. Bit(s) setzen
F40D  4F          LD    C,A        Flags wieder zurück
F40E  F1          POP   AF           Formatstringparameter
F40F  F1          POP   AF           (für Anfang) löschen
F410  CD 1B F4    CALL  F41B       Formatstring weiter auswerten
F413  7C          LD    A,H        Vorkomma-
F414  85          ADD   L          und Nachkommastellen addieren
F415  FE 15       CP    15         Gesamtstellenzahl>=21 ?
```

```
F417  D2 4B F3     JP     NC,F34B        dann "Improper argument"
F41A  C9           RET

*********************************** Formatstring weiter auswerten
                                    IN/OUT: DE: Formatstringzeiger
                                             B: restl. Stringlänge
                                             H: Vorkommastellenzahl
                                             C: Formatierungsflags
                                    OUT: L: Nachkommastellenzahl
F41B  2E 00        LD     L,00           Nachkommastellenzahl=0
F41D  04           INC    B
F41E  05           DEC    B                Formatstringlänge=0 ?
F41F  C8           RET    Z              dann fertig
F420  1A           LD     A,(DE)         Zeichen aus Formatstring
F421  FE 2E        CP     2E             "." ?
F423  28 14        JR     Z,F439         dann Nachkommastellen
F425  FE 2C        CP     2C             "," ?
F427  28 0A        JR     Z,F433         dann Komma-Einteilungen
F429  FE 23        CP     23             "#" ?
F42B  20 15        JR     NZ,F442        nein ? dann "^^^^" prüfen
F42D  24           INC    H              Vorkommastellenzahl erhöhen
F42E  13           INC    DE               nächstes
F42F  05           DEC    B                Zeichen
F430  20 EE        JR     NZ,F420        noch kein Ende ? dann weiter
F432  C9           RET
F433  79           LD     A,C            Formatierungsflags
F434  F6 02        OR     02             Bit f. Komma-Einteilung setzen
F436  4F           LD     C,A
F437  18 F4        JR     F42D           Vorkommastellenzahl erhöhen
F439  2C           INC    L              Nachkommastellenzahl erhöhen
F43A  13           INC    DE               nächstes
F43B  05           DEC    B                Zeichen
F43C  C8           RET    Z              Ende ? dann fertig
F43D  1A           LD     A,(DE)         Zeichen aus Formatstring
F43E  FE 23        CP     23             "#" ?
F440  28 F7        JR     Z,F439         dann weitere Nachkommastellen
F442  EB           EX     DE,HL          Stellenzahlen nach DE
F443  E5           PUSH   HL             Formatstringzeiger retten
F444  FE 5E        CP     5E
F446  20 18        JR     NZ,F460
F448  23           INC    HL
F449  BE           CP     (HL)             nicht "^^^^" ?
F44A  20 14        JR     NZ,F460          dann keine
F44C  23           INC    HL               Exponentialdarstellung
F44D  BE           CP     (HL)
F44E  20 10        JR     NZ,F460
F450  23           INC    HL
F451  BE           CP     (HL)
F452  20 0C        JR     NZ,F460
F454  23           INC    HL
F455  78           LD     A,B            Stringlänge
F456  D6 04        SUB    04             minus 4 für "^^^^"
F458  38 06        JR     C,F460         Länge zu klein ? d. k. "^^^^"
F45A  47           LD     B,A            restliche Länge
F45B  E3           EX     (SP),HL        Stringzeiger retten
F45C  79           LD     A,C              Flag für
F45D  F6 40        OR     40               Exponentialdarstellung
F45F  4F           LD     C,A              setzen
```

```
F460  E1        POP   HL        Stringzeiger nach "^^^^"
F461  EB        EX    DE,HL     wieder nach DE
F462  78        LD    A,B       restliche Stringlänge
F463  B7        OR    A         =0 ?
F464  C8        RET   Z         dann fertig
F465  79        LD    A,C         Flag für Vorzeichen
F466  E6 08     AND   08          vor der Zahl ?
F468  C0        RET   NZ        dann zurück
F469  1A        LD    A,(DE)    Zeichen aus Formatstring
F46A  FE 2D     CP    2D        "-" ?
F46C  3E 10     LD    A,10      Flag für Vorz. nach der Zahl
F46E  28 06     JR    Z,F476    ggf. setzen
F470  1A        LD    A,(DE)    Zeichen aus Formatstring
F471  FE 2B     CP    2B        "+" ?
F473  C0        RET   NZ        nein ? dann zurück
F474  3E 18     LD    A,18      Vorz. n. d. Z./pos. Vorz.="+"
F476  B1        OR    C         bisherige Flags setzen
F477  4F        LD    C,A       Formatierungsflags
F478  13        INC   DE          nächstes
F479  05        DEC   B           Zeichen
F47A  C9        RET

********************************** Basic-Befehl WRITE
F47B  CD C6 C1  CALL  C1C6        opt. Streamnr. holen/setzen
F47E  F5        PUSH  AF          alte Streamnummer retten
F47F  CD 51 DD  CALL  DD51        Statementende ?
F482  38 39     JR    C,F4BD      dann Linefeed ausgeben
F484  CD FB CE  CALL  CEFB        Ausdruck holen
F487  F5        PUSH  AF          nächstes Zeichen
F488  E5        PUSH  HL          und Basic-PC retten
F489  CD 45 FF  CALL  FF45        Typflag des FAC holen
F48C  28 0B     JR    Z,F499      String ?
F48E  CD 8F EE  CALL  EE8F        sonst FAC nach ASCII wandeln
F491  CD DC F7  CALL  F7DC        String auf Stringstack
F494  CD 28 F8  CALL  F828        String ausgeben, vom Stack
F497  18 0D     JR    F4A6
F499  3E 22     LD    A,22        '"'
F49B  CD 56 C3  CALL  C356        ausgeben
F49E  CD 28 F8  CALL  F828        String ausgeben, vom Stack
F4A1  3E 22     LD    A,22        '"'
F4A3  CD 56 C3  CALL  C356        ausgeben
F4A6  E1        POP   HL          Basic-PC
F4A7  F1        POP   AF          Zeichen
F4A8  28 13     JR    Z,F4BD      Statementende ? dann Linefeed
F4AA  FE 3B     CP    3B          ";" ?
F4AC  28 05     JR    Z,F4B3      dann "," ausgeben
F4AE  FE 2C     CP    2C          "," ?
F4B0  C2 46 F3  JP    NZ,F346     nein ? dann "Syntax error"
F4B3  CD 3F DD  CALL  DD3F        nächstes Zeichen
F4B6  3E 2C     LD    A,2C        ","
F4B8  CD 56 C3  CALL  C356        ausgeben
F4BB  18 C7     JR    F484        nächsten Ausdruck holen
F4BD  CD 4E C3  CALL  C34E        Linefeed ausgeben
F4C0  F1        POP   AF          alte Streamnr.
F4C1  C3 A2 C1  JP    C1A2        wieder setzen
```

```
*********************************** RAM-Zeiger initialisieren
                                    IN : DE: LoRAM-Zeiger
                                         HL: HiRAM-Zeiger
                                    OUT: CY=1 für kein Platz
F4C4  01 00 AC   LD    BC,AC00        Zeiger auf Basic-Systembereich
F4C7  CD BE FF   CALL  FFBE           mit HiRAM vergleichen
F4CA  D0         RET   NC             HiRAM nicht kleiner ? d. Fehl.
F4CB  22 7B AE   LD    (AE7B),HL      HiRAM als Himem
F4CE  22 8F B0   LD    (B08F),HL      als Ende der Strings
F4D1  22 7D AE   LD    (AE7D),HL      als Ende des freien RAMs
F4D4  EB         EX    DE,HL
F4D5  22 7F AE   LD    (AE7F),HL      LoRAM als Start d. freien RAMs
F4D8  01 2F 01   LD    BC,012F        Platz für Token-Buffer
F4DB  09         ADD   HL,BC          addieren
F4DC  D8         RET   C              kein Platz für Buffer ?
F4DD  22 81 AE   LD    (AE81),HL      Basic-Programmstart setzen
F4E0  EB         EX    DE,HL
F4E1  23         INC   HL             HiRAM+1
F4E2  B7         OR    A              CY=0
F4E3  ED 52      SBC   HL,DE          minus Programmstart
F4E5  D8         RET   C              kein Platz ?
F4E6  7C         LD    A,H            Größe des freien Bereichs
F4E7  FE 04      CP    04             <=$0400 ?
F4E9  D8         RET   C              dann Fehler
F4EA  AF         XOR   A              Null, CY=0
F4EB  32 91 B0   LD    (B091),A       Kassettenbuffer nicht reserv.
F4EE  C9         RET

*********************************** Basic-Befehl MEMORY
F4EF  CD 3E FC   CALL  FC3E           Garbage collection
F4F2  CD 91 CE   CALL  CE91           Adresse holen, nach DE
F4F5  E5         PUSH  HL             Basic-PC retten
F4F6  CD 50 F7   CALL  F750           HIMEM-Zeiger setzen
F4F9  CD 75 F6   CALL  F675           Kassettenbuffer ggf. freigeben
F4FC  22 7B AE   LD    (AE7B),HL      HIMEM-Zeiger setzen
F4FF  E1         POP   HL             Basic-PC zurück
F500  C9         RET

*********************************** Test auf Platz für Binärdatei
                                    IN : DE: Startadresse
                                         BC: Länge
F501  D5         PUSH  DE             Startadresse
F502  2A 7F AE   LD    HL,(AE7F)      LoRAM-Zeiger
F505  EB         EX    DE,HL          nach DE
F506  2A 7B AE   LD    HL,(AE7B)      HIMEM
F509  CD CF FF   CALL  FFCF           -LoRAM=Größe d. Basic-Bereichs
F50C  E3         EX    (SP),HL        retten, Startadresse zurück
F50D  CD CF FF   CALL  FFCF           -LoRAM
F510  D1         POP   DE             Größe des Basic-Bereichs
F511  13         INC   DE             Größe f. Byte danach
F512  CD B8 FF   CALL  FFB8           mit anderer Differenz vergl.
F515  38 03      JR    C,F51A         Startadr. innerhalb Basic-B. ?
F517  2B         DEC   HL             Korr., da Start+Länge=Ende+1
F518  09         ADD   HL,BC          Länge addieren
F519  D0         RET   NC             Ende nicht im Basic-Bereich ?
F51A  C3 3E F7   JP    F73E           sonst "Memory full"
```

```
**********************************  Größe des Stringbereichs holen
                                    OUT: BC: Größe
F51D  D5          PUSH  DE
F51E  E5          PUSH  HL
F51F  2A 8D B0    LD    HL,(B08D)     Zeiger auf Start der Strings
F522  EB          EX    DE,HL         nach DE
F523  2A 8F B0    LD    HL,(B08F)     Zeiger auf Ende der Strings
F526  CD DA FF    CALL  FFDA          Differenz nach BC
F529  E1          POP   HL
F52A  D1          POP   DE
F52B  C9          RET

**********************************  Programm/Var.-Zeiger korrigieren
                                    IN : BC: Offset
F52C  2A 83 AE    LD    HL,(AE83)
F52F  09          ADD   HL,BC         Offset zu Programmende
F530  22 83 AE    LD    (AE83),HL
F533  2A 85 AE    LD    HL,(AE85)
F536  09          ADD   HL,BC         und Variablenstart addieren
F537  22 85 AE    LD    (AE85),HL

**********************************  Arrayzeiger korrigieren
                                    IN : BC: Offset
F53A  2A 87 AE    LD    HL,(AE87)
F53D  09          ADD   HL,BC         Offset zu Start
F53E  22 87 AE    LD    (AE87),HL
F541  2A 89 AE    LD    HL,(AE89)
F544  09          ADD   HL,BC         und Ende der Felder addieren
F545  22 89 AE    LD    (AE89),HL
F548  C9          RET

**********************************  Variablen in Stringbereich retten
                                    OUT: BC: Länge des Variablenber.
                                         DE: Länge der einfachen Var.
F549  2A 85 AE    LD    HL,(AE85)     Zeiger auf Variablenstart
F54C  EB          EX    DE,HL         nach DE
F54D  2A 87 AE    LD    HL,(AE87)     Zeiger auf Start der Felder
F550  CD CF FF    CALL  FFCF          -Variablenstart
F553  E5          PUSH  HL            gibt Länge der einfachen Var.
F554  2A 89 AE    LD    HL,(AE89)     Zeiger auf Ende der Arrays
F557  CD DA FF    CALL  FFDA          -Variablenstart
F55A  C5          PUSH  BC            gibt Länge des Variablenber.
F55B  2A 8D B0    LD    HL,(B08D)     Zeiger auf Start der Strings
F55E  EB          EX    DE,HL         nach DE, als neue Var.-Adresse
F55F  2A 89 AE    LD    HL,(AE89)     Zeiger auf Ende der Arrays
F562  2B          DEC   HL            Zeiger auf letztes Array-Byte
F563  78          LD    A,B             Länge<>0 ?
F564  B1          OR    C
F565  C4 F5 FF    CALL  NZ,FFF5       d. Var. unter Strings versch.
F568  EB          EX    DE,HL         Zeiger auf verschobene Var.
F569  22 8D B0    LD    (B08D),HL     als neuen Start der Strings
F56C  C1          POP   BC              Längen
F56D  D1          POP   DE              zurück
F56E  C3 B1 D5    JP    D5B1          Variablenbereich freigeben
```

```
***********************************    Variablen a. Stringbereich zurück
                                       IN : BC: Länge d. Var.-Bereichs
                                            DE: Länge der einfachen Var.
F571  2A 83 AE     LD     HL,(AE83)       Zeiger auf Programmende
F574  22 85 AE     LD     (AE85),HL       als Zeiger auf Variablenstart
F577  EB           EX     DE,HL           nach DE
F578  19           ADD    HL,DE           Länge d. einf. Var. addieren
F579  22 87 AE     LD     (AE87),HL       gibt Zeiger auf Arraystart
F57C  2A 8D B0     LD     HL,(B08D)       Zeiger auf Start der Strings
F57F  23           INC    HL              1. benutztes Stringbyte
F580  78           LD     A,B               Länge<>0 ?
F581  B1           OR     C
F582  C4 F2 FF     CALL   NZ,FFF2         dann Variablen zurückholen
F585  2B           DEC    HL              Zeiger vor 1. ben. Stringbyte
F586  22 8D B0     LD     (B08D),HL       als neuen Start der Strings
F589  EB           EX     DE,HL           Zeiger auf Ende aller Var.
F58A  22 89 AE     LD     (AE89),HL       als Ende der Arrays setzen
F58D  C9           RET

***********************************    Basic-Stackpointer initialisieren
F58E  F5           PUSH   AF
F58F  E5           PUSH   HL
F590  21 8B AE     LD     HL,AE8B         Basic-Stackpointer
F593  22 8B B0     LD     (B08B),HL       neu setzen
F596  3E 01        LD     A,01            1 Byte
F598  CD B0 F5     CALL   F5B0            auf Basic-Stack reservieren
F59B  36 00        LD     (HL),00         Null als Stackende auf Stack
F59D  E1           POP    HL
F59E  F1           POP    AF
F59F  C9           RET

***********************************    Eintrag vom Basic-Stack holen
                                       IN : A: Größe des Eintrags
                                       OUT: HL: Zeiger auf Eintrag
F5A0  2A 8B B0     LD     HL,(B08B)       Basic-Stackpointer
F5A3  2F           CPL                      Zweierkomplement bilden
F5A4  3C           INC    A                 (für Offset nach unten)
F5A5  C8           RET    Z               Größe=0 ? dann fertig
F5A6  85           ADD    L
F5A7  6F           LD     L,A               zu Basic-Stackpointer
F5A8  3E FF        LD     A,FF              addieren
F5AA  8C           ADC    H                 (entspricht Subtraktion
F5AB  67           LD     H,A               der Eintragsgröße)

***********************************    Basic-Stackpointer neu setzen
                                       IN : HL: neuer Basic-Stackpointer
F5AC  22 8B B0     LD     (B08B),HL
F5AF  C9           RET

***********************************    Platz auf Basic-Stack reservieren
                                       IN : A: benötigter Platz
                                       OUT: HL: Zeiger auf Platz
F5B0  2A 8B B0     LD     HL,(B08B)       Basic-Stackpointer
F5B3  E5           PUSH   HL              retten
F5B4  85           ADD    L
F5B5  6F           LD     L,A               benötigte Platzgröße
F5B6  8C           ADC    H                 zu Basic-Stackpointer
F5B7  95           SUB    L                 addieren
```

```
F5B8  67          LD    H,A
F5B9  22 8B B0    LD    (B08B),HL     Basic-Stackpointer neu setzen
F5BC  3E 78       LD    A,78            $4F78 addieren
F5BE  85          ADD   L               entspr. $B088 subtrahieren
F5BF  3E 4F       LD    A,4F            ($B088=Ende des Basic-Stack-
F5C1  8C          ADC   H               bereichs)
F5C2  E1          POP   HL            alter SP=Zeiger a. neuen Platz
F5C3  D0          RET   NC            keine Bereichsüberschreitung ?
F5C4  CD 8E F5    CALL  F58E          sonst SP neu initialisieren
F5C7  C3 3E F7    JP    F73E          "Memory full"

***********************************  Stringbereich löschen
F5CA  2A 8F B0    LD    HL,(B08F)     Zeiger auf Ende der Strings
F5CD  22 8D B0    LD    (B08D),HL     als Start der Strings setzen
F5D0  C9          RET

***********************************  Stringbereich-Platz reservieren
                                    IN : A: ben. Größe des Platzes
                                    OUT: DE: Zeiger auf neuen Platz
F5D1  2F          CPL                   Zweierkomplement
F5D2  4F          LD    C,A             der Platzgröße
F5D3  06 FF       LD    B,FF            als Offset
F5D5  03          INC   BC              nach BC
F5D6  CD E6 F5    CALL  F5E6          Stringbereich erweitern
F5D9  D0          RET   NC            genügend Platz ?
F5DA  CD 3E FC    CALL  FC3E          sonst Garbage collection
F5DD  CD E6 F5    CALL  F5E6          Stringbereich erweitern
F5E0  D0          RET   NC            genügend Platz ?
F5E1  1E 0E       LD    E,0E          Nr. für "String space full"
F5E3  C3 94 CA    JP    CA94          Fehler ausgeben

***********************************  Stringbereich erweitern
                                    IN : BC: negativer Offset
                                    OUT: DE: Zeiger a. 1. freies Byte
                                            CY=1 für kein Platz
F5E6  2A 89 AE    LD    HL,(AE89)     Zeiger auf Ende der Felder
F5E9  EB          EX    DE,HL         nach DE
F5EA  2A 8D B0    LD    HL,(B08D)     Zeiger auf Start der Strings
F5ED  09          ADD   HL,BC         +Offset=neuer Stringstart
F5EE  CD B8 FF    CALL  FFB8          mit Ende d. Felder vergleichen
F5F1  D8          RET   C             kein Platz ?
F5F2  22 8D B0    LD    (B08D),HL     sonst Start der Strings setzen
F5F5  23          INC   HL            Zeiger auf 1. freies Byte
F5F6  EB          EX    DE,HL         nach DE
F5F7  C9          RET

***********************************  Platz für Programm/Var. schaffen
                                    IN/OUT: DE: Einfügeadresse
                                            BC: benötigter Platz
                                    OUT:    HL: neues Ende der Arrays
F5F8  2A 89 AE    LD    HL,(AE89)     Zeiger auf Ende der Arrays
F5FB  C5          PUSH  BC
F5FC  D5          PUSH  DE
F5FD  D5          PUSH  DE
F5FE  E5          PUSH  HL
F5FF  CD 18 F6    CALL  F618          auf Platz prüfen
F602  DA 3E F7    JP    C,F73E        kein Platz ? dann Fehler
F605  E1          POP   HL            Zeiger auf Ende der Arrays
```

```
F606  C1          POP   BC          Einfügeadresse
F607  D5          PUSH  DE          neues Ende der Arrays
F608  7D          LD    A,L           Ende der Arrays
F609  91          SUB   C             -Einfügeadresse
F60A  4F          LD    C,A           gibt Länge des zu
F60B  7C          LD    A,H           verschiebenden Bereichs,
F60C  98          SBC   B             nach BC
F60D  47          LD    B,A
F60E  2B          DEC   HL          letztes Array-Byte
F60F  1B          DEC   DE          letztes Byte des Platzes
F610  B1          OR    C           Länge <>0 ?
F611  C4 F5 FF    CALL  NZ,FFF5     dann verschieben
F614  E1          POP   HL          neues Ende der Arrays
F615  D1          POP   DE          Zeiger auf Platz
F616  C1          POP   BC          Länge des Platzes
F617  C9          RET

************************************ auf Platz prüfen
                                    IN : HL: Ende des belegten Ber.
                                         BC: benötigter Platz
                                    OUT: DE: neues Ende des bel. Ber.
                                          CY=1, wenn kein Platz
F618  09          ADD   HL,BC         Platz addieren
F619  D8          RET   C             Übertrag ? dann kein Platz
F61A  EB          EX    DE,HL         neues Ende des bel. Ber. n. DE
F61B  CD 22 F6    CALL  F622          Überschn. m. Stringbereich ?
F61E  D0          RET   NC            nein ? dann o.k.
F61F  CD 3E FC    CALL  FC3E          sonst Garbage collection
F622  2A 8D B0    LD    HL,(B08D)     Zeiger auf Start der Strings
F625  C3 B8 FF    JP    FFB8          CY=1, wenn DE im Stringbereich

************************************ Größe des freien Speicherplatzes
                                    OUT: HL: Größe in Bytes
F628  2A 89 AE    LD    HL,(AE89)     Zeiger auf Ende der Arrays
F62B  EB          EX    DE,HL         nach DE
F62C  2A 8D B0    LD    HL,(B08D)     Zeiger auf Start der Strings
F62F  C3 CF FF    JP    FFCF          -Ende der Arrays=freier Platz

************************************ Eingabebuffer belegen
                                    OUT: DE: Zeiger auf Buffer
F632  11 01 00    LD    DE,0001       Offset und Flag f. Eingabebuf.
F635  18 03       JR    F63A

************************************ Ausgabebuffer belegen
                                    OUT: DE: Zeiger auf Buffer
F637  11 02 08    LD    DE,0802       Offset und Flag f. Ausgabebuf.
F63A  C5          PUSH  BC
F63B  E5          PUSH  HL
F63C  21 91 B0    LD    HL,B091       Zeiger auf Buffer-Flags
F63F  7E          LD    A,(HL)        Buffer-Flags
F640  B7          OR    A
F641  20 1D       JR    NZ,F660       Buffer schon reserviert ?
F643  D5          PUSH  DE            Offset und Flag für Buffer
F644  E5          PUSH  HL            Zeiger auf Flags
F645  21 00 10    LD    HL,1000       benötigter Platz für 2 Buffer
F648  01 00 00    LD    BC,0000       min. Bufferadresse
F64B  CD 43 F7    CALL  F743          Platz reservieren
F64E  22 92 B0    LD    (B092),HL     Zeiger vor Platz
```

```
F651  EB          EX     DE,HL
F652  2A 7D AE    LD     HL,(AE7D)     altes Ende des freien RAMs
F655  22 94 B0    LD     (B094),HL     retten
F658  EB          EX     DE,HL         Zeiger vor reserviertem Platz
F659  22 7D AE    LD     (AE7D),HL     als neues Ende des freien RAMs
F65C  E1          POP    HL            Zeiger auf Flags
F65D  D1          POP    DE            Offset und Flag für Buffer
F65E  3E 04       LD     A,04          Flag für I/O-Buffer reserviert
F660  B3          OR     E             Flag f. Ein- bzw. Ausgabebuf.
F661  77          LD     (HL),A        Flags wieder speichern
F662  2A 92 B0    LD     HL,(B092)     Zeiger vor Buffer
F665  23          INC    HL            Zeiger auf Buffer
F666  1E 00       LD     E,00          Offset lo=0
F668  19          ADD    HL,DE         Buffer-Offset addieren
F669  EB          EX     DE,HL         Bufferadresse nach DE
F66A  E1          POP    HL
F66B  C1          POP    BC
F66C  C9          RET

*********************************** Eingabebuffer freigeben
F66D  3E FE       LD     A,FE            Maske für Eingabebuffer
F66F  18 06       JR     F677

*********************************** Ausgabebuffer freigeben
F671  3E FD       LD     A,FD            Maske für Ausgabebuffer
F673  18 02       JR     F677

*********************************** Ein-/Ausgabebuffer ggf. freigeben
F675  3E FF       LD     A,FF            Flag f. Bufferstatus n. verän.
F677  C5          PUSH   BC
F678  D5          PUSH   DE
F679  E5          PUSH   HL
F67A  21 91 B0    LD     HL,B091         Zeiger auf Buffer-Flags
F67D  A6          AND    (HL)            entsprechende Flags löschen
F67E  77          LD     (HL),A          und wieder speichern
F67F  FE 04       CP     04              Flag f. reserv., aber unben. ?
F681  20 16       JR     NZ,F699         nein ? dann zurück
F683  2A 92 B0    LD     HL,(B092)       Zeiger vor Bufferbereich
F686  EB          EX     DE,HL           nach DE
F687  21 00 10    LD     HL,1000         Länge der Buffer
F68A  CD 2E F7    CALL   F72E            Bereich ggf. freigeben
F68D  20 0A       JR     NZ,F699         nicht freigegeben ?
F68F  AF          XOR    A                 sonst Flag für kein Buffer-
F690  32 91 B0    LD     (B091),A          bereich reserviert setzen
F693  2A 94 B0    LD     HL,(B094)       altes Ende des freien RAMs
F696  22 7D AE    LD     (AE7D),HL       wieder setzen
F699  E1          POP    HL
F69A  D1          POP    DE
F69B  C1          POP    BC
F69C  C9          RET

*********************************** Basic-Befehl SYMBOL
F69D  FE 80       CP     80              folgt Token für AFTER ?
F69F  28 2C       JR     Z,F6CD          dann SYMBOL AFTER
F6A1  CD 67 CE    CALL   CE67            Byte-Ausdruck holen
F6A4  4F          LD     C,A             als Nr. des Zeichens nach C
F6A5  CD 37 DD    CALL   DD37            Test auf Komma
F6A8  2C                                 ","
```

```
F6A9  06 08        LD     B,08          Zähler für 8 Matrixbytes
F6AB  CD 67 CE     CALL   CE67          Byte holen
F6AE  F5           PUSH   AF            auf Stack
F6AF  05           DEC    B
F6B0  28 08        JR     Z,F6BA        keine weiteren Matrixbytes ?
F6B2  CD 55 DD     CALL   DD55          folgt Komma ?
F6B5  38 F4        JR     C,F6AB        dann nächstes Matrixbyte
F6B7  AF           XOR    A             sonst Defaultwert 0
F6B8  18 F4        JR     F6AE
F6BA  EB           EX     DE,HL         Basic-PC nach DE
F6BB  79           LD     A,C           Nr. des Zeichens
F6BC  CD A5 BB     CALL   BBA5          Adresse d. zugeh. Matrix holen
F6BF  30 68        JR     NC,F729       außerhalb User-Matrix ?
F6C1  01 08 00     LD     BC,0008       Zähler für Matrixbytes
F6C4  09           ADD    HL,BC         zu Startadresse addieren
F6C5  F1           POP    AF            Matrixbyte
F6C6  2B           DEC    HL
F6C7  77           LD     (HL),A        speichern
F6C8  0D           DEC    C
F6C9  20 FA        JR     NZ,F6C5       weitere Matrixbytes ?
F6CB  EB           EX     DE,HL         Basic-PC wieder nach HL
F6CC  C9           RET

***********************************   Basic-Befehl SYMBOL AFTER
F6CD  CD 3F DD     CALL   DD3F          AFTER-Token übergehen
F6D0  CD 86 CE     CALL   CE86          Integerwert holen
F6D3  E5           PUSH   HL            Basic-PC retten
F6D4  21 00 01     LD     HL,0100       Maximalwert
F6D7  CD B8 FF     CALL   FFB8          mit Integerwert vergleichen
F6DA  38 4D        JR     C,F729        Wert zu groß ?
F6DC  D5           PUSH   DE            Wert retten
F6DD  CD AE BB     CALL   BBAE          Param. d. exist. Usermatrix h.
F6E0  EB           EX     DE,HL         Adresse der User-Matrix n. DE
F6E1  30 1D        JR     NC,F700       keine User-Matrix definiert ?
F6E3  2F           CPL                    Zweierkomplement
F6E4  6F           LD     L,A             des 1. Zeichens
F6E5  26 00        LD     H,00            der User-Matrix
F6E7  23           INC    HL              nach HL
F6E8  29           ADD    HL,HL
F6E9  29           ADD    HL,HL           mal 8, da 8 Bytes/Zeichen,
F6EA  29           ADD    HL,HL           gibt Größe der Matrix
F6EB  1B           DEC    DE            Adresse der User-Matrix -1
F6EC  CD 2E F7     CALL   F72E          User-Matrix-Bereich freigeben
F6EF  20 38        JR     NZ,F729       nicht freigegeben ? d. Fehler
F6F1  2A 96 B0     LD     HL,(B096)     altes Ende des freien RAMs
F6F4  22 7D AE     LD     (AE7D),HL     wieder setzen
F6F7  CD 75 F6     CALL   F675          Ein/Ausgabebuffer ggf. freig.
F6FA  11 00 01     LD     DE,0100       Kennzeichen für keine Matrix
F6FD  CD AB BB     CALL   BBAB          User-Matrix zurücksetzen
F700  D1           POP    DE            Integerw. (Nr. d. 1. Zeichens)
F701  CD 06 F7     CALL   F706          User-Matrix neu setzen
F704  E1           POP    HL            Basic-PC zurück
F705  C9           RET

***********************************   User-Matrix neu setzen
                                        IN : DE: Nr. des 1. Zeichens
F706  AF           XOR    A
F707  93           SUB    E               $100-Nr. des 1. Zeichens
```

```
F708  6F          LD    L,A             gibt Zahl der Zeichen
F709  3E 01       LD    A,01            in User-Matrix,
F70B  9A          SBC   D               nach HL
F70C  67          LD    H,A
F70D  B5          OR    L             =0 ?
F70E  C8          RET   Z             dann keine User-Matrix
F70F  D5          PUSH  DE            Nr. des 1. Zeichens retten
F710  29          ADD   HL,HL           Zahl der Zeichen mal 8
F711  29          ADD   HL,HL           gibt Größe der User-Matrix
F712  29          ADD   HL,HL
F713  01 00 40    LD    BC,4000       min. Adresse für User-Matrix
F716  CD 43 F7    CALL  F743          Platz reservieren
F719  EB          EX    DE,HL         Zeiger vor Platz nach DE
F71A  2A 7D AE    LD    HL,(AE7D)     altes Ende des freien RAMs
F71D  22 96 B0    LD    (B096),HL     retten
F720  EB          EX    DE,HL         Zeiger vor freien Platz
F721  22 7D AE    LD    (AE7D),HL     als neues Ende des freien RAMs
F724  D1          POP   DE            Nr. des 1. Zeichens d. Matrix
F725  23          INC   HL            Zeiger auf neue User-Matrix
F726  C3 AB BB    JP    BBAB          setzen

F729  1E 05       LD    E,05          Nr. für "Improper argument"
F72B  C3 94 CA    JP    CA94          Fehler ausgeben

**********************************  Speicherbereich freigeben
                                    IN : DE: Adresse d. Bereichs -1
                                         HL: Größe des Bereichs
                                    OUT: Z=1 für Bereich freigegeben
F72E  E5          PUSH  HL            Größe retten
F72F  2A 7B AE    LD    HL,(AE7B)     HIMEM-Zeiger
F732  CD B8 FF    CALL  FFB8          =Startadresse des Bereichs ?
F735  E1          POP   HL
F736  C0          RET   NZ            nein ? dann zurück
F737  19          ADD   HL,DE         Länge addieren
F738  22 7D AE    LD    (AE7D),HL     als neues Ende des freien RAMs
F73B  EB          EX    DE,HL         nach DE
F73C  18 12       JR    F750          als HIMEM setzen

F73E  1E 07       LD    E,07          Nr. für "Memory full"
F740  C3 94 CA    JP    CA94          Fehler ausgeben

**********************************  Platz im Speicher reservieren
                                    IN : HL: Größe des Platzes
                                         BC: min. Adresse für Platz
                                    OUT: HL: neuer HIMEM-Zeiger
                                             (=Zeiger auf Platz-1)
                                         Z=1, CY=0
F743  EB          EX    DE,HL         ben. Größe des Bereichs n. DE
F744  2A 7B AE    LD    HL,(AE7B)     HIMEM-Zeiger
F747  CD CF FF    CALL  FFCF          Größe abz., gibt neues HIMEM
F74A  CD BE FF    CALL  FFBE          mit min. Adresse vergleichen
F74D  38 EF       JR    C,F73E        zu klein ? dann Fehler
F74F  EB          EX    DE,HL         neuen HIMEM-Zeiger nach DE

**********************************  HIMEM neu setzen
                                    IN : DE: neuer HIMEM-Zeiger (464)
                                         DE: neuer HIMEM-Zeiger+1 (664/6128)
                                    OUT: HL: wie DE, IN
```

```
F750  CD 3E FC     CALL   FC3E        Garbage collection
F753  D5           PUSH   DE          neuen HIMEM-Zeiger retten
F754  2A 7D AE     LD     HL,(AE7D)   Zeiger auf Ende des freien RAM
F757  CD B8 FF     CALL   FFB8        kleiner als neuer HIMEM-Zg. ?
F75A  38 E2        JR     C,F73E      dann "Memory full"
F75C  CD 1D F5     CALL   F51D        Größe des Stringbereichs n. BC
F75F  2A 89 AE     LD     HL,(AE89)   Zeiger auf Ende der Felder
F762  09           ADD    HL,BC       Größe des Stringber. addieren
F763  38 D9        JR     C,F73E      Übertrag ? dann "Memory full"
F765  2B           DEC    HL          letztes nach Versch. ben. Byte
F766  CD B8 FF     CALL   FFB8        kleiner als neues HIMEM ?
F769  30 D3        JR     NC,F73E     sonst "Memory full"
F76B  2A 7B AE     LD     HL,(AE7B)   alter HIMEM-Zeiger
F76E  EB           EX     DE,HL       nach DE, neuer nach HL
F76F  CD CF FF     CALL   FFCF        Differenz als Offset nach HL
F772  22 98 B0     LD     (B098),HL   Offset für Verschiebung setzen
F775  11 BB F7     LD     DE,F7BB     Routine für Offset addieren
F778  CD 74 DA     CALL   DA74        sämtl. Stringvar. durchgehen
F77B  ED 4B 98 B0  LD     BC,(B098)   Offset
F77F  78           LD     A,B
F780  07           RLCA               Offset negativ ?
F781  38 16        JR     C,F799      dann nach unten verschieben
F783  B1           OR     C           Offset=0 ?
F784  28 2F        JR     Z,F7B5      dann fertig
F786  2A 8F B0     LD     HL,(B08F)   Zeiger auf Ende der Strings
F789  54           LD     D,H           als Quellendadresse
F78A  5D           LD     E,L           nach DE
F78B  09           ADD    HL,BC       Offset addieren, gibt Zieladr.
F78C  E5           PUSH   HL          als neues Stringende retten
F78D  CD 1D F5     CALL   F51D        Größe des Stringber. als Länge
F790  EB           EX     DE,HL       altes Ende n. HL, neues n. DE
F791  78           LD     A,B
F792  B1           OR     C             Länge <>0 ?
F793  C4 F5 FF     CALL   NZ,FFF5     dann Stringbereich verschieben
F796  E1           POP    HL          neues Ende der Strings
F797  18 15        JR     F7AE        Zeiger setzen
F799  2A 8D B0     LD     HL,(B08D)   Zeiger auf Start der Strings
F79C  54           LD     D,H           als Quellstartadresse
F79D  5D           LD     E,L           nach DE
F79E  09           ADD    HL,BC       Offset addieren, gibt Zieladr.
F79F  E5           PUSH   HL          als neuen Stringstart retten
F7A0  CD 1D F5     CALL   F51D        Größe des Stringber. als Länge
F7A3  EB           EX     DE,HL       alten Start n. HL, neuen n. DE
F7A4  23           INC    HL            Zeiger auf 1. belegtes Byte
F7A5  13           INC    DE            des Stringbereichs
F7A6  78           LD     A,B
F7A7  B1           OR     C             Länge <>0 ?
F7A8  C4 F2 FF     CALL   NZ,FFF2     dann Stringbereich verschieben
F7AB  EB           EX     DE,HL       Zieladresse nach HL
F7AC  2B           DEC    HL          -1 gibt neues Stringende
F7AD  D1           POP    DE          neuen Start der Strings
F7AE  22 8F B0     LD     (B08F),HL     Ende
F7B1  EB           EX     DE,HL         und
F7B2  22 8D B0     LD     (B08D),HL     Start der Strings setzen
F7B5  E1           POP    HL          neuen HIMEM-Zeiger
F7B6  22 7B AE     LD     (AE7B),HL   setzen
F7B9  AF           XOR    A           CY=1, Z=0
F7BA  C9           RET
```

```
*********************************** Offset zu Stringadresse addieren
                                    IN : DE: Zeiger Descriptor-Ende
                                            BC: Stringadresse
                                            ($B098): Offset
F7BB  2A 83 AE    LD    HL,(AE83)     Zeiger auf Programmende
F7BE  CD BE FF    CALL  FFBE          mit Stringadresse vergleichen
F7C1  D0          RET   NC            String im Programm ? d. fertig
F7C2  2A 98 B0    LD    HL,(B098)     Offset
F7C5  09          ADD   HL,BC         zu Stringadresse addieren
F7C6  EB          EX    DE,HL         neue Adresse nach DE
F7C7  72          LD    (HL),D          Adresse in
F7C8  2B          DEC   HL              Stringdescriptor
F7C9  73          LD    (HL),E          eintragen
F7CA  C9          RET

*********************************** String überlesen, auf Stringstack
                                    IN : HL: Zeiger vor String
                                    OUT: HL: Zeiger nach String
                                            B: Stringlänge
F7CB  23          INC   HL            Zeiger auf String
F7CC  CD F9 F7    CALL  F7F9          String nach Routine auf Stack
F7CF  7E          LD    A,(HL)        Zeichen aus String
F7D0  FE 22       CP    22            '"' ?
F7D2  CA 3F DD    JP    Z,DD3F        d. fertig, nächst. Zeichen h.
F7D5  B7          OR    A             Zeilenende ?
F7D6  28 37       JR    Z,F80F        dann Sonderzeichen eliminieren
F7D8  04          INC   B             Stringlänge
F7D9  23          INC   HL            und -zeiger erhöhen
F7DA  18 F3       JR    F7CF          weiter prüfen

*********************************** String bis Zl.-Ende übl., auf St.
                                    IN : HL: Zeiger auf String
                                    OUT: HL: Zeiger auf Zeilenende
                                            B: Stringlänge
F7DC  CD F9 F7    CALL  F7F9          String nach Routine auf Stack
F7DF  7E          LD    A,(HL)        Zeichen aus String
F7E0  B7          OR    A             Zeilenende ?
F7E1  C8          RET   Z             dann fertig
F7E2  23          INC   HL            Stringzeiger
F7E3  04          INC   B             und -länge erhöhen
F7E4  18 F9       JR    F7DF          weiter prüfen

*********************************** String bis Trennzeich. übernehmen
                                    IN : HL: Zeiger auf String
                                            A: Trennzeichen
                                    OUT: HL: Zeiger auf Stringende
F7E6  CD F9 F7    CALL  F7F9          String nach Routine auf Stack
F7E9  4F          LD    C,A           Trennzeichen
F7EA  7E          LD    A,(HL)        Zeichen aus String
F7EB  B7          OR    A             Zeilenende ?
F7EC  28 21       JR    Z,F80F
F7EE  B9          CP    C             Trennzeichen ?
F7EF  28 1E       JR    Z,F80F
F7F1  FE 2C       CP    2C            "," ?
F7F3  28 1A       JR    Z,F80F
F7F5  23          INC   HL            sonst Stringzeiger
F7F6  04          INC   B             und -länge erhöhen
F7F7  18 F1       JR    F7EA          weiter prüfen
```

```
***********************************     Routine weiterf., String a. Stack
                                        IN : HL: Zeiger auf String
                                        OUT: HL: Zeiger auf Stringende
                                             B: Länge
F7F9  D1          POP    DE                Aufrufadresse
F7FA  E5          PUSH   HL                Zeiger auf String
F7FB  06 00       LD     B,00              Zähler für Länge
F7FD  CD FB FF    CALL   FFFB              aufrufende Rout. weiterführen
F800  D1          POP    DE                Zeiger auf String
F801  E5          PUSH   HL                Zeiger auf Stringende retten
F802  21 BA B0    LD     HL,B0BA           Zeiger für Stringdescriptor
F805  70          LD     (HL),B            Länge
F806  23          INC    HL
F807  73          LD     (HL),E            und Adresse des Strings
F808  23          INC    HL                in Descriptor eintragen
F809  72          LD     (HL),D
F80A  CD BA FB    CALL   FBBA              Descr. auf Stack, Zg. n. FAC
F80D  E1          POP    HL                Zeiger auf Stringende
F80E  C9          RET

***********************************     Sonderzeichen am Stringende elim.
                                        IN : HL: Zeiger auf Stringende
                                             B: Stringlänge
                                        OUT: B: neue Länge
F80F  E5          PUSH   HL                Zeiger auf Stringende
F810  04          INC    B
F811  05          DEC    B                 Länge
F812  28 12       JR     Z,F826            =0 ? dann fertig
F814  2B          DEC    HL
F815  7E          LD     A,(HL)            Zeichen aus String
F816  FE 20       CP     20                Space ?
F818  28 F7       JR     Z,F811            dann eliminieren
F81A  FE 09       CP     09                TAB ?
F81C  28 F3       JR     Z,F811            dann eliminieren
F81E  FE 0D       CP     0D                CR ?
F820  28 EF       JR     Z,F811            dann eliminieren
F822  FE 0A       CP     0A                LF ?
F824  28 EB       JR     Z,F811            dann eliminieren
F826  E1          POP    HL                Zeiger auf Stringende zurück
F827  C9          RET

***********************************     String vom Stringstack, ausgeben
F828  CD DA FB    CALL   FBDA              String im FAC v. Stack löschen
F82B  C8          RET    Z                 Länge=0 ? dann zurück
F82C  1A          LD     A,(DE)            Zeichen aus String
F82D  13          INC    DE
F82E  CD 6E C3    CALL   C36E              ausgeben
F831  10 F9       DJNZ   F82C              weitere Zeichen ?
F833  C9          RET

***********************************     Basic-Funktion LOWER$
F834  01 39 F8    LD     BC,F839           Routine Kleinschrift forcieren
F837  18 0C       JR     F845

***********************************     auf Kleinschrift forcieren
                                        IN/OUT: A: Zeichen
F839  FE 41       CP     41                <"A" ?
F83B  D8          RET    C                 dann zurück
```

```
F83C  FE 5B     CP    5B          >="Z"+1 ?
F83E  D0        RET   NC          dann zurück
F83F  C6 20     ADD   20          sonst nach Kleinschrift
F841  C9        RET

********************************** Basic-Funktion UPPER$
F842  01 8A FF  LD    BC,FF8A     Routine Großschrift forcieren
F845  C5        PUSH  BC          Routinenadresse retten
F846  2A C2 B0  LD    HL,(B0C2)   Zeiger auf Descriptor
F849  7E        LD    A,(HL)      Stringlänge
F84A  CD 19 FC  CALL  FC19        Platz für neuen String reserv.
F84D  D5        PUSH  DE          Zeiger auf Platz f. neuen Str.
F84E  CD DA FB  CALL  FBDA        alten String vom Stringstack
F851  E1        POP   HL          Zeiger für neuen String
F852  C1        POP   BC          Routinenadresse
F853  3C        INC   A           Länge ausgleichen
F854  3D        DEC   A           Länge
F855  CA BA FB  JP    Z,FBBA      schon alle Zeichen ?
F858  F5        PUSH  AF          restl. Länge retten
F859  1A        LD    A,(DE)      Zeichen aus String
F85A  13        INC   DE
F85B  CD F9 FF  CALL  FFF9        Wandlungsroutine ausführen
F85E  77        LD    (HL),A      und in neuen String speichern
F85F  23        INC   HL
F860  F1        POP   AF          restl. Länge zurück
F861  18 F1     JR    F854        weiter wandeln

********************************** Stringverknüpfung "+"
                                  IN : HL: Zeiger auf 1. Descriptor
                                       2. Descriptor im FAC
F863  E5        PUSH  HL
F864  7E        LD    A,(HL)      Länge des 1. Strings
F865  2A C2 B0  LD    HL,(B0C2)   Zeiger auf 2. Descriptor
F868  86        ADD   (HL)        Länge des 2. Strings addieren
F869  1E 0F     LD    E,0F        Nr. für "String too long"
F86B  DA 94 CA  JP    C,CA94      Gesamtlänge >$FF ? dann Fehler
F86E  CD 19 FC  CALL  FC19        Platz für String reservieren
F871  E1        POP   HL          Zeiger auf 1. Descriptor
F872  D5        PUSH  DE          Zeiger auf Platz f. neuen Str.
F873  E5        PUSH  HL          Zeiger auf 1. Descriptor
F874  CD DA FB  CALL  FBDA        2. String vom Stringstack lö.
F877  48        LD    C,B         Länge des 2. Strings
F878  EB        EX    DE,HL       Zeiger auf 2. String nach HL
F879  E3        EX    (SP),HL     retten, Zg. 1. Descriptor zur.
F87A  CD E8 FB  CALL  FBE8        1. String vom Stringstack lö.
F87D  E1        POP   HL            zweitobersten Stackeintrag
F87E  E3        EX    (SP),HL       (Zeiger auf neuen String)
F87F  78        LD    A,B         Länge des 1. Strings
F880  CD 8B F8  CALL  F88B        1. String kopieren
F883  D1        POP   DE          Zeiger auf 2. String
F884  79        LD    A,C         Länge des 2. Strings
F885  CD 8B F8  CALL  F88B        2. String kopieren
F888  C3 BA FB  JP    FBBA        Gesamtstring auf Stringstack
```

```
**********************************   String kopieren
                                     IN : A: Länge
                                          DE: Quelladresse;
                                          HL: Zieladresse
                                     OUT: HL: Zeiger nach kop. String
F88B  C5        PUSH  BC
F88C  EB        EX    DE,HL            Quelle n. DE, Ziel n. HL
F88D  4F        LD    C,A              Länge lo
F88E  06 00     LD    B,00             Länge hi=0
F890  B7        OR    A                Länge <>0 ?
F891  C4 F2 FF  CALL  NZ,FFF2          dann kopieren
F894  EB        EX    DE,HL            Quelle und Ziel wieder zurück
F895  C1        POP   BC
F896  C9        RET

**********************************   Stringvergleich
                                     IN : HL: Zeiger auf 1. Descriptor
                                          Zg. 2. Descriptor im FAC
                                     OUT: A: Vergleichsergebnis
                                          A=$00, Z=1 für gleich
                                          A=$01, CY=0 f. Str1 < Str2
                                          A=$FF, CY=1 f. Str1 > Str2
F897  E5        PUSH  HL               Zeiger auf 1. Descriptor
F898  CD DA FB  CALL  FBDA             2. String vom Stringstack
F89B  48        LD    C,B              Länge des 2. Strings
F89C  E1        POP   HL               Zeiger auf 1. Descriptor
F89D  D5        PUSH  DE               Zeiger auf 2. String
F89E  CD E8 FB  CALL  FBE8             1. String vom Stringstack
F8A1  E1        POP   HL               Zeiger auf 2. String
F8A2  78        LD    A,B              1. Stringlänge
F8A3  B1        OR    C                2. Stringlänge
F8A4  C8        RET   Z                beide =0 ? dann Strings gleich
F8A5  79        LD    A,C              2. Stringlänge
F8A6  B7        OR    A                =0 ?
F8A7  28 0C     JR    Z,F8B5           dann 1. String größer
F8A9  78        LD    A,B              1. Stringlänge
F8AA  B7        OR    A                =0 ?
F8AB  28 09     JR    Z,F8B6           dann 2. String größer
F8AD  05        DEC   B                  Stringlängen
F8AE  0D        DEC   C                  herunterzählen
F8AF  1A        LD    A,(DE)           Zeichen aus 1. String
F8B0  13        INC   DE
F8B1  BE        CP    (HL)             mit Zeichen aus 2. Str. vergl.
F8B2  23        INC   HL
F8B3  28 ED     JR    Z,F8A2           gleich ? dann nächstes Zeichen
F8B5  3F        CCF                    CY=1, wenn 1. String größer
F8B6  9F        SBC   A                A=$FF, wenn 1. String größer
F8B7  C0        RET   NZ               1. String größer ?
F8B8  3C        INC   A                sonst A=1
F8B9  C9        RET

**********************************   Basic-Funktion BIN$
F8BA  CD CE F8  CALL  F8CE             Ausdruck und Stellenzahl holen
F8BD  D5        PUSH  DE               Basic-PC retten
F8BE  CD 14 F1  CALL  F114             Zahl nach ASCII wandeln
F8C1  EB        EX    DE,HL            Adresse des Strings nach HL
F8C2  18 5E     JR    F922             String in Stringbereich/-stack
```

```
********************************** Basic-Funktion HEX$
F8C4  CD CE F8    CALL  F8CE       Ausdruck und Stellenzahl holen
F8C7  D5          PUSH  DE         Basic-PC retten
F8C8  CD 19 F1    CALL  F119       Zahl nach ASCII wandeln
F8CB  EB          EX    DE,HL      Adresse des Strings nach HL
F8CC  18 54       JR    F922       String in Stringbereich/-stack

********************************** Ausdruck und Stellenzahl holen
                                   IN : HL: Basic-PC
                                   OUT: B: Stellenzahl; C: Typ des Ausdrucks
                                        HL: Zeiger auf Ausdruck; DE: Basic-PC
F8CE  CD FB CE    CALL  CEFB       Ausdruck holen
F8D1  CD 53 FF    CALL  FF53       und auf Basic-Stack, Typ n. C
F8D4  CD 55 DD    CALL  DD55       Test auf Komma
F8D7  9F          SBC   A          A=0, wenn kein Komma (Default)
F8D8  DC 67 CE    CALL  C,CE67     ggf. Byte holen, als Stellenz.
F8DB  FE 11       CP    11         >=17 ?
F8DD  D2 9C FA    JP    NC,FA9C    dann "Improper argument"
F8E0  47          LD    B,A        Stellenzahl nach B
F8E1  CD 37 DD    CALL  DD37       Test auf Klammer zu
F8E4  29                           "("
F8E5  EB          EX    DE,HL      Basic-PC nach DE
F8E6  79          LD    A,C        Typ des Ausdrucks
F8E7  C3 A0 F5    JP    F5A0       Ausdruck wieder v. Basic-Stack

********************************** Basic-Funktion DEC$
F8EA  CD 37 DD    CALL  DD37       Test auf Klammer auf (??)
F8ED  28                           "("
F8EE  CD FB CE    CALL  CEFB       Ausdruck holen
F8F1  CD 37 DD    CALL  DD37       Test auf Komma
F8F4  2C                           ","
F8F5  CD 53 FF    CALL  FF53       FAC auf Basic-Stack retten
F8F8  CD 9F CE    CALL  CE9F       Stringausdruck holen, v. Stack
F8FB  CD 37 DD    CALL  DD37       Test auf Klammer zu
F8FE  29                           ")"
F8FF  E5          PUSH  HL         Basic-PC retten
F900  79          LD    A,C        Typ des 1. Ausdrucks
F901  CD A0 F5    CALL  F5A0       Ausdruck wieder v. Basic-Stack
F904  D5          PUSH  DE         Zeiger auf String
F905  79          LD    A,C        Typ des 1. Ausdrucks
F906  CD 4B FF    CALL  FF4B       Ausdruck in FAC kopieren
F909  D1          POP   DE         Zeiger auf String
F90A  78          LD    A,B        Stringlänge
F90B  B7          OR    A          <>0 ?
F90C  C4 BA F3    CALL  NZ,F3BA    dann Formatstring auswerten
F90F  30 0A       JR    NC,F91B    Fehler in der Auswertung ?
F911  78          LD    A,B        restliche Stringlänge
F912  B7          OR    A          <>0 ?
F913  20 06       JR    NZ,F91B    dann Fehler
F915  79          LD    A,C        Formatierungsflags
F916  CD 9F EE    CALL  EE9F       Zahl nach ASCII wandeln
F919  18 07       JR    F922       String in Stringbereich/-stack

F91B  C3 9C FA    JP    FA9C       "Improper argument"

********************************** Basic-Funktion STR$
F91E  E5          PUSH  HL         Basic-PC retten
F91F  CD 9D EE    CALL  EE9D       FAC nach ASCII wandeln
```

```
F922  E5          PUSH   HL          Zeiger auf String retten
F923  01 FF FF    LD     BC,FFFF     Stringlänge =-1
F926  03          INC    BC          Länge erhöhen
F927  7E          LD     A,(HL)      Zeichen aus String
F928  23          INC    HL
F929  B7          OR     A           kein Ende ?
F92A  20 FA       JR     NZ,F926     dann weiter prüfen
F92C  79          LD     A,C         Länge lo
F92D  CD 19 FC    CALL   FC19        Platz in Stringbereich reserv.
F930  E1          POP    HL          Zeiger auf String
F931  B7          OR     A           Länge
F932  D5          PUSH   DE          Zeiger auf Platz im Stringber.
F933  C4 F2 FF    CALL   NZ,FFF2     Länge <>0 ? dann String kop.
F936  D1          POP    DE          Zeiger auf kopierten String
F937  CD BA FB    CALL   FBBA        auf Stringstack
F93A  E1          POP    HL          Basic-PC zurück
F93B  C9          RET

*********************************** Basic-Funktion LEFT$
F93C  CD E9 F9    CALL   F9E9          String und Byte holen
F93F  0E 00       LD     C,00          Startposition
F941  18 2A       JR     F96D          Teilstring holen

*********************************** Basic-Funktion RIGHT$
F943  CD E9 F9    CALL   F9E9          String und Byte holen
F946  1A          LD     A,(DE)        Stringlänge
F947  90          SUB    B             minus Bytewert
F948  4F          LD     C,A           gibt Startposition
F949  18 22       JR     F96D          Teilstring holen

*********************************** Basic-Funktion MID$
F94B  CD 37 DD    CALL   DD37          Test auf Klammer auf
F94E  28                               "("
F94F  CD E9 F9    CALL   F9E9          String und Byte holen
F952  78          LD     A,B           Byte
F953  B7          OR     A             =0 ?
F954  CA 9C FA    JP     Z,FA9C        dann "Improper argument"
F957  05          DEC    B             Byte -1
F958  48          LD     C,B           als Startpos. f. Teilstring
F959  D5          PUSH   DE            Descriptorzeiger
F95A  C5          PUSH   BC            und Startposition retten
F95B  CD FB F9    CALL   F9FB          2. Byte als Länge holen
F95E  C1          POP    BC            Startposition zurück
F95F  E3          EX     (SP),HL       PC retten, Descr.-Zeiger zur.
F960  7E          LD     A,(HL)        Länge des Strings
F961  91          SUB    C             minus Startposition
F962  06 00       LD     B,00          Länge 0
F964  38 05       JR     C,F96B        Startposition zu groß ?
F966  BB          CP     E             mit gewünschter Länge vergl.
F967  47          LD     B,A           restliche Stringlänge
F968  38 01       JR     C,F96B        gew. Länge zu groß ?
F96A  43          LD     B,E           sonst gewünschte Länge setzen
F96B  EB          EX     DE,HL         Descriptorzeiger nach DE
F96C  E1          POP    HL            Basic-PC zurück
F96D  CD 37 DD    CALL   DD37          Test auf Klammer zu
F970  29                               ")"
```

```
*********************************** Teilstring holen
                                    IN : DE: Zeiger auf Descriptor
                                         C: Startposition
                                         B: gewünschte Länge
                                    OUT: Descriptorzeiger im FAC
F971  E5          PUSH   HL
F972  EB          EX     DE,HL         Descriptorzeiger nach HL
F973  7E          LD     A,(HL)        Länge des Strings
F974  B8          CP     B             mit gewünschter Länge vergl.
F975  78          LD     A,B           gewünschte Länge
F976  30 03       JR     NC,F97B       nicht zu groß ?
F978  7E          LD     A,(HL)          sonst Stringlänge als Länge
F979  0E 00       LD     C,00            und Startposition=0
F97B  F5          PUSH   AF            Länge retten
F97C  CD 19 FC    CALL   FC19          Platz im Stringbereich reserv.
F97F  D5          PUSH   DE            Zeiger auf reservierten Platz
F980  CD E8 FB    CALL   FBE8          alten String vom Stringstack
F983  EB          EX     DE,HL         Zeiger auf String nach HL
F984  D1          POP    DE            Zeiger für neuen String
F985  06 00       LD     B,00          Startposition hi=0
F987  09          ADD    HL,BC         Startposition addieren
F988  F1          POP    AF            gewünschte Länge
F989  4F          LD     C,A           nach C
F98A  B7          OR     A             Länge <>0 ?
F98B  C4 F2 FF    CALL   NZ,FFF2       dann String kopieren
F98E  CD BA FB    CALL   FBBA          String auf St.-Stack u. in FAC
F991  E1          POP    HL
F992  C9          RET
*********************************** Basic-Befehl MID$
F993  CD 37 DD    CALL   DD37          Test auf Klammer auf
F996  28                               "("
F997  CD 86 D6    CALL   D686          Variable holen, ggf. neu anl.
F99A  CD 3C FF    CALL   FF3C          Test auf Stringvariable
F99D  E5          PUSH   HL            Basic-PC retten
F99E  EB          EX     DE,HL         Var.-Adr. (des Descr.) n. HL
F99F  CD 21 FB    CALL   FB21          String in Stringber. forcieren
F9A2  E3          EX     (SP),HL       Descr.-Zeiger retten, PC zur.
F9A3  CD 37 DD    CALL   DD37          Test auf Komma
F9A6  2C                               ","
F9A7  CD 6D CE    CALL   CE6D          Bytewert <>0 holen
F9AA  47          LD     B,A           als Startposition nach B
F9AB  CD FB F9    CALL   F9FB          2. Bytewert holen
F9AE  4B          LD     C,E           als Länge nach C
F9AF  CD 37 DD    CALL   DD37          Test auf Klammer zu
F9B2  29                               ")"
F9B3  CD 37 DD    CALL   DD37          Test auf "="
F9B6  EF                               Token für "="
F9B7  C5          PUSH   BC            Parameter retten
F9B8  CD 9F CE    CALL   CE9F          String holen, vom Stringstack
F9BB  78          LD     A,B           Stringlänge
F9BC  C1          POP    BC            Parameter zurück
F9BD  E3          EX     (SP),HL       PC retten, Var.-Descr.-Z. zur.
F9BE  0C          INC    C             gewünschte Länge
F9BF  0D          DEC    C
F9C0  28 25       JR     Z,F9E7        =0 ? dann fertig
F9C2  F5          PUSH   AF            Länge des Stringausdrucks
F9C3  7E          LD     A,(HL)        Länge des Strings in Variable
F9C4  90          SUB    B             - Startposition
```

```
F9C5  DA 9C FA    JP     C,FA9C       Startpos. außerhalb String ?
F9C8  3C          INC    A            +1 = restl. Länge nach Startp.
F9C9  B9          CP     C            m. zu ersetzender Länge vergl.
F9CA  38 01       JR     C,F9CD       restl. Stringlänge zu klein ?
F9CC  79          LD     A,C          sonst übergebenen Längenwert
F9CD  4F          LD     C,A          zu ersetzende Länge
F9CE  78          LD     A,B          Startposition
F9CF  3D          DEC    A            -1 = Offset zu Stringanfang
F9D0  23          INC    HL           Descr.-Zeiger auf Stringadr.
F9D1  86          ADD    (HL)           Adresse aus Descriptor
F9D2  23          INC    HL             + Offset
F9D3  66          LD     H,(HL)         ergibt Startadresse,
F9D4  6F          LD     L,A            nach HL
F9D5  8C          ADC    H
F9D6  95          SUB    L
F9D7  67          LD     H,A
F9D8  F1          POP    AF           Länge des Stringausdrucks
F9D9  47          LD     B,A          nach B
F9DA  EB          EX     DE,HL        Startadresse nach DE
F9DB  79          LD     A,C          zu ersetzende Länge
F9DC  B8          CP     B            mit Stringausdrucklänge vergl.
F9DD  38 01       JR     C,F9E0       zu ersetzende Länge kleiner ?
F9DF  78          LD     A,B          sonst Länge des Stringausdr.
F9E0  4F          LD     C,A          als zu ersetzende Länge
F9E1  06 00       LD     B,00         Länge hi=0
F9E3  B7          OR     A            Länge <>0 ?
F9E4  C4 F2 FF    CALL   NZ,FFF2      dann Teilstring ersetzen
F9E7  E1          POP    HL           Basic-PC zurück
F9E8  C9          RET

*********************************** String und Byte holen
                                    OUT: DE: Zeiger auf Descriptor
                                         A,B: Byte
                                         beim CPC 664/6128:
                                         CY=0
                                         Z=1, wenn Byte=0
F9E9  CD A5 CE    CALL   CEA5           Stringausdruck holen
F9EC  CD 37 DD    CALL   DD37           Test auf Komma
F9EF  2C                                ","
F9F0  E5          PUSH   HL             Basic-PC retten
F9F1  2A C2 B0    LD     HL,(B0C2)      Zeiger auf Descriptor
F9F4  E3          EX     (SP),HL        retten, PC zurück
F9F5  CD 67 CE    CALL   CE67           Byteausdruck holen
F9F8  47          LD     B,A            Bytewert nach B
F9F9  D1          POP    DE             Zeiger auf Descriptor
F9FA  C9          RET

*********************************** 2. Byte für MID$ holen
                                    OUT: E: Byte
F9FB  1E FF       LD     E,FF           Default-Wert
F9FD  7E          LD     A,(HL)         folgendes Zeichen
F9FE  FE 29       CP     29             Klammer zu ?
FA00  C8          RET    Z              dann Default-Wert
FA01  CD 37 DD    CALL   DD37           Test auf Komma
FA04  2C                                ","
FA05  CD 67 CE    CALL   CE67           Bytewert holen
FA08  5F          LD     E,A            nach E
FA09  C9          RET
```

```
*********************************** Basic-Funktion LEN
FA0A  CD DA FB     CALL   FBDA       String v. Stack, Länge nach A
FA0D  C3 0A FF     JP     FF0A       Länge in FAC eintragen

*********************************** Basic-Funktion ASC
FA10  CD 70 FA     CALL   FA70       1. Zeichen aus String nach A
FA13  C3 0A FF     JP     FF0A       und in FAC eintragen

*********************************** Basic-Funktion CHR$
FA16  CD 92 FA     CALL   FA92       FAC nach Byte wandeln
FA19  F5           PUSH   AF         Byte retten
FA1A  3E 01        LD     A,01       Stringlänge =1
FA1C  CD 19 FC     CALL   FC19       Platz für String reservieren
FA1F  F1           POP    AF         Bytewert
FA20  12           LD     (DE),A     als 1. Zeichen in String
FA21  C3 BA FB     JP     FBBA       String auf Stringstack

*********************************** Basic-Funktion INKEY$
FA24  E5           PUSH   HL         Basic-PC retten
FA25  CD 2A FA     CALL   FA2A       Funktion ausführen
FA28  E1           POP    HL         Basic-PC
FA29  C9           RET

*********************************** String für INKEY$ holen
FA2A  CD 39 C4     CALL   C439       Taste lesen
FA2D  38 EA        JR     C,FA19     gedrückt ? d. in String wand.
FA2F  AF           XOR    A          sonst Länge Null
FA30  32 BA B0     LD     (B0BA),A   in Stringdescriptor setzen
FA33  C3 BA FB     JP     FBBA       String auf Stringstack

*********************************** Basic-Funktion STRING$
FA36  CD 67 CE     CALL   CE67       Bytewert holen
FA39  4F           LD     C,A        als  Länge
FA3A  CD 37 DD     CALL   DD37       Test auf Komma
FA3D  2C                             ","
FA3E  CD FB CE     CALL   CEFB       Ausdruck holen
FA41  CD 37 DD     CALL   DD37       Test auf Klammer zu
FA44  29                             ")"
FA45  E5           PUSH   HL         Basic-PC retten
FA46  CD 45 FF     CALL   FF45       Typflag holen
FA49  28 05        JR     Z,FA50     String ?
FA4B  CD 92 FA     CALL   FA92       sonst FAC nach Byte wandeln
FA4E  18 03        JR     FA53
FA50  CD 70 FA     CALL   FA70       1. Zeichen aus String holen
FA53  41           LD     B,C        Länge
FA54  4F           LD     C,A        Zeichen
FA55  18 07        JR     FA5E       String generieren

*********************************** Basic-Funktion SPACE$
FA57  CD 92 FA     CALL   FA92       FAC nach Byte wandeln
FA5A  47           LD     B,A        Byte als Länge nach B
FA5B  0E 20        LD     C,20       Space
FA5D  E5           PUSH   HL         Basic-PC retten
FA5E  78           LD     A,B        Länge
FA5F  CD 19 FC     CALL   FC19       Platz für String reservieren
FA62  04           INC    B          Länge korrigieren
FA63  05           DEC    B          Zähler
FA64  28 05        JR     Z,FA6B     =0 ? dann fertig
```

```
FA66  79          LD     A,C           Zeichen
FA67  12          LD     (DE),A        in String speichern
FA68  13          INC    DE
FA69  18 F8       JR     FA63
FA6B  CD BA FB    CALL   FBBA          String auf Stringstack
FA6E  E1          POP    HL            Basic-PC zurück
FA6F  C9          RET

********************************** 1. Zeichen aus String holen
                                    OUT: A: Zeichen
FA70  CD DA FB    CALL   FBDA          String vom Stringstack
FA73  28 27       JR     Z,FA9C        Länge =0 ? dann Fehler
FA75  1A          LD     A,(DE)        1. Zeichen aus String
FA76  C9          RET

********************************** Basic-Funktion VAL
FA77  CD DA FB    CALL   FBDA          String vom Stringstack
FA7A  CA 0A FF    JP     Z,FF0A        Länge=0 ? dann FAC=0
FA7D  EB          EX     DE,HL         Zeiger auf String nach HL
FA7E  E5          PUSH   HL            und retten
FA7F  5F          LD     E,A           Länge lo
FA80  16 00       LD     D,00          Länge hi=0
FA82  19          ADD    HL,DE         Länge addieren
FA83  5E          LD     E,(HL)        Zeichen nach String
FA84  72          LD     (HL),D        durch Null für Ende ersetzen
FA85  E3          EX     (SP),HL       Zg. Stringende r., Str.-Zg. z.
FA86  D5          PUSH   DE            Zeichen nach String
FA87  CD A3 EC    CALL   ECA3          String nach binär wandeln
FA8A  D1          POP    DE            Zeichen nach String
FA8B  E1          POP    HL            Zeiger nach String
FA8C  73          LD     (HL),E        Zeichen wieder setzen
FA8D  D8          RET    C             kein Fehler bei Wandlung ?
FA8E  1E 0D       LD     E,0D          Nr. für "Type mismatch"
FA90  18 0C       JR     FA9E          Fehler ausgeben

********************************** FAC nach Byte wandeln
                                    OUT: A: Byte
FA92  E5          PUSH   HL            Basic-PC retten
FA93  CD 8D FE    CALL   FE8D          CINT, FAC nach Integer nach HL
FA96  EB          EX     DE,HL         Integerwert nach DE
FA97  E1          POP    HL            Basic-PC
FA98  7A          LD     A,D           Integerwert hi
FA99  B7          OR     A
FA9A  7B          LD     A,E           Integerwert lo
FA9B  C8          RET    Z             Hi-Byte=0 ? dann o.k.
FA9C  1E 05       LD     E,05          Nr. für "Improper argument"
FA9E  C3 94 CA    JP     CA94          Fehler ausgeben

********************************** Basic-Funktion INSTR
FAA1  CD FB CE    CALL   CEFB          Ausdruck holen
FAA4  CD 45 FF    CALL   FF45          Typflag holen
FAA7  0E 01       LD     C,01          Default-Startposition
FAA9  28 0F       JR     Z,FABA        Stringausdruck ? dann Default
FAAB  CD 92 FA    CALL   FA92          sonst FAC nach Byte wandeln
FAAE  B7          OR     A             Bytewert
FAAF  CA 9C FA    JP     Z,FA9C        =0 ? dann "Improper argument"
FAB2  4F          LD     C,A           sonst Byte als Startposition
FAB3  CD 37 DD    CALL   DD37          Test auf Komma
```

```
FAB6  2C                               ","
FAB7  CD A5 CE   CALL   CEA5           Stringausdruck holen
FABA  CD 37 DD   CALL   DD37           Test auf Komma
FABD  2C                               ","
FABE  E5         PUSH   HL             Basic-PC retten
FABF  2A C2 B0   LD     HL,(B0C2)      Zeiger auf Descriptor
FAC2  E3         EX     (SP),HL        retten, PC zurück
FAC3  CD 9F CE   CALL   CE9F           Suchstring holen, vom Stack
FAC6  CD 37 DD   CALL   DD37           Test auf Klammer zu
FAC9  29                               "("
FACA  E3         EX     (SP),HL        PC retten, Descr.-Zeiger zur.
FACB  79         LD     A,C            Startposition
FACC  CD D4 FA   CALL   FAD4           Suchstring im 1. String suchen
FACF  CD 0A FF   CALL   FF0A           Ergebnis nach FAC
FAD2  E1         POP    HL             Basic-PC zurück
FAD3  C9         RET

***********************************  Suchstring in String suchen
                                     IN : A: Start-Suchposition
                                          B: Suchstringlänge
                                          DE: Suchstringadresse
                                          HL: Descriptorzg. des Str.,
                                              in dem zu suchen ist
                                     OUT: Ergebnis(-position)
FAD4  F5         PUSH   AF             Startposition retten
FAD5  48         LD     C,B            Länge des Suchstrings
FAD6  D5         PUSH   DE             Adresse des Suchstrings
FAD7  CD E8 FB   CALL   FBE8           1. String vom Stack, Params h.
FADA  E1         POP    HL             Adresse des Suchstrings
FADB  F1         POP    AF             Startposition
FADC  E5         PUSH   HL             Adresse des Suchstrings
FADD  6F         LD     L,A            Startposition
FADE  60         LD     H,B            Länge des 1. Strings
FADF  78         LD     A,B
FAE0  BD         CP     L              mit Startposition vergleichen
FAE1  38 2D      JR     C,FB10         Startp.>Länge ? d. nicht gef.
FAE3  2D         DEC    L              Startposition-1 = Offset
FAE4  7D         LD     A,L              Offset zu Adresse
FAE5  83         ADD    E                des 1. Strings addieren,
FAE6  5F         LD     E,A              gibt Startadresse
FAE7  8A         ADC    D                für
FAE8  93         SUB    E                Suche,
FAE9  57         LD     D,A              nach DE
FAEA  78         LD     A,B            Länge des 1. Strings
FAEB  95         SUB    L              Offset  abziehen
FAEC  47         LD     B,A            restliche Länge
FAED  79         LD     A,C            Länge des Suchstrings
FAEE  D6 01      SUB    01             CY=1, wenn Länge =0
FAF0  7D         LD     A,L            Offset
FAF1  3C         INC    A              +1=Position für Ergebnis
FAF2  38 1D      JR     C,FB11         Suchstringlänge=0 ? d. gefund.
FAF4  E3         EX     (SP),HL        1. Stringl. r., Suchstr.-Adr.
FAF5  C5         PUSH   BC
FAF6  D5         PUSH   DE             Suchadresse im 1. String
FAF7  E5         PUSH   HL             Suchstringadresse
FAF8  1A         LD     A,(DE)         Zeichen aus 1. String
FAF9  BE         CP     (HL)           =Zeichen aus Suchstring ?
FAFA  20 0D      JR     NZ,FB09        nein ? dann nächste Position
```

```
FAFC  23          INC   HL          Zeiger
FAFD  0D          DEC   C           und Zähler für Suchstring
FAFE  28 13       JR    Z,FB13      Suchstring abgearb. ? d. gef.
FB00  13          INC   DE          Zeiger
FB01  05          DEC   B           und Zähler für 1. String
FB02  20 F4       JR    NZ,FAF8     1. String nicht zu Ende ?
FB04  E1          POP   HL          Suchstringadresse
FB05  D1          POP   DE          Suchadresse im 1. String
FB06  C1          POP   BC          Stringlängen
FB07  18 07       JR    FB10        Flag für nicht gefunden setzen
FB09  E1          POP   HL          Suchstringadresse
FB0A  D1          POP   DE          Suchadresse im 1. String
FB0B  C1          POP   BC          Stringlängen
FB0C  13          INC   DE          Zeiger
FB0D  05          DEC   B           und Zähler für 1. String
FB0E  20 E5       JR    NZ,FAF5     ggf. ab nächster Position su.
FB10  AF          XOR   A           Flag für nicht gefunden
FB11  D1          POP   DE          Länge des 1. String löschen
FB12  C9          RET
FB13  E1          POP   HL          Suchstringadresse
FB14  D1          POP   DE          Suchadresse im 1. String
FB15  C1          POP   BC          Stringlängen
FB16  E1          POP   HL          Länge des 1. Strings
FB17  7C          LD    A,H         nach A
FB18  90          SUB   B           - restl. Suchlänge
FB19  3C          INC   A           +1 gibt gefundene Position
FB1A  C9          RET

*********************************** Strings in Stringber. forcieren
FB1B  11 2E FB    LD    DE,FB2E        Zeiger auf Routine
FB1E  C3 74 DA    JP    DA74           sämtliche Strings durchgehen

*********************************** String in Stringber. forcieren
                                    IN : HL: Zeiger auf Descriptor
FB21  E5          PUSH  HL             Zeiger auf Descriptor
FB22  7E          LD    A,(HL)         Länge
FB23  23          INC   HL
FB24  4E          LD    C,(HL)         und Stringadresse
FB25  23          INC   HL             aus Descriptor laden
FB26  46          LD    B,(HL)
FB27  EB          EX    DE,HL
FB28  B7          OR    A              Länge <>0 ?
FB29  C4 2E FB    CALL  NZ,FB2E        dann in Stringber. forcieren
FB2C  E1          POP   HL             Zeiger auf Descriptor
FB2D  C9          RET

*********************************** String in Stringber. forcieren
                                    IN : BC: Stringadresse
                                         DE: Descriptoradresse +2
FB2E  2A 8D B0    LD    HL,(B08D)      Zeiger auf Start der Strings
FB31  CD BE FF    CALL  FFBE           mit Stringadresse vergleichen
FB34  30 07       JR    NC,FB3D        Str. nicht im Stringbereich ?
FB36  2A 8F B0    LD    HL,(B08F)      Zeiger auf Ende der Strings
FB39  CD BE FF    CALL  FFBE           mit Stringadresse vergleichen
FB3C  D0          RET   NC             String im Stringbereich ?
FB3D  EB          EX    DE,HL          Descriptorzeiger+2 nach HL
FB3E  2B          DEC   HL               -2 gibt Zeiger
FB3F  2B          DEC   HL               auf Stringdescriptor
```

```
FB40  E5          PUSH  HL          retten
FB41  CD 8F FB    CALL  FB8F        String in Stringber. kopieren
FB44  EB          EX    DE,HL       Zeiger auf neuen Descr. n. DE
FB45  E1          POP   HL          Zeiger auf alten Descriptor
FB46  C3 A6 FB    JP    FBA6        neuen Descr. in alten kopieren

***********************************  Descriptor ggf. auf Stringstack
FB49  2A C2 B0    LD    HL,(B0C2)   Zeiger auf Descriptor
FB4C  11 BA B0    LD    DE,B0BA     Zg. auf Ende des Stringstacks
FB4F  CD B8 FF    CALL  FFB8        Descriptor auf Stringstack ?
FB52  D8          RET   C           dann o.k.
FB53  CD 8F FB    CALL  FB8F        String in Stringber. kopieren
FB56  C3 BA FB    JP    FBBA        und Descriptor auf Stringstack

***********************************  String kopieren, vom Stringstack
                                    IN : String im FAC
                                    OUT: A: Stringlänge
                                         DE: Zeiger auf String
                                         HL: Zeiger auf Descriptor
FB59  2A C2 B0    LD    HL,(B0C2)   Zeiger auf Descriptor
FB5C  E5          PUSH  HL          retten
FB5D  7E          LD    A,(HL)      Stringlänge
FB5E  B7          OR    A
FB5F  28 26       JR    Z,FB87      =0 ? dann fertig
FB61  23          INC   HL
FB62  5E          LD    E,(HL)      Stringadresse aus Descriptor
FB63  23          INC   HL          laden, nach DE
FB64  56          LD    D,(HL)
FB65  2A 81 AE    LD    HL,(AE81)   Zeiger auf Programmstart
FB68  CD B8 FF    CALL  FFB8        String unterhalb Programm ?
FB6B  30 1E       JR    NC,FB8B     dann in Stringber. kopieren
FB6D  2A 8F B0    LD    HL,(B08F)   Zeiger auf Ende der Strings
FB70  CD B8 FF    CALL  FFB8        String oberhalb Stringber. ?
FB73  38 16       JR    C,FB8B      dann in Stringber. kopieren
FB75  2A 83 AE    LD    HL,(AE83)   Zeiger auf Programmende
FB78  CD B8 FF    CALL  FFB8        String im Programm ?
FB7B  30 0A       JR    NC,FB87     dann nicht kopieren
FB7D  E1          POP   HL          Zeiger auf Descriptor
FB7E  E5          PUSH  HL
FB7F  11 9C B0    LD    DE,B09C     Zeiger auf 1. Stringstackelem.
FB82  CD B8 FF    CALL  FFB8        Descr. 1. Element im Stack ?
FB85  20 04       JR    NZ,FB8B     nein ? dann String kopieren
FB87  E1          POP   HL          Zeiger auf Descriptor
FB88  C3 FF FB    JP    FBFF        String vom Stringstack

FB8B  E1          POP   HL          Zeiger auf Descriptor
FB8C  CD FF FB    CALL  FBFF        String vom Stringstack

***********************************  String in Stringbereich kopieren
                                    IN : HL: Zeiger auf Descriptor
                                    OUT: HL: neuer Zeiger auf Descr.
                                         DE: Zeiger auf String
FB8F  7E          LD    A,(HL)      Länge
FB90  CD 19 FC    CALL  FC19        Platz für neuen String reserv.
FB93  D5          PUSH  DE          Zeiger auf neuen Platz
FB94  4E          LD    C,(HL)      Stringlänge
FB95  06 00       LD    B,00        Länge hi=0
FB97  23          INC   HL
```

```
FB98  7E           LD     A,(HL)         Stringadresse
FB99  23           INC    HL             aus Descriptor
FB9A  66           LD     H,(HL)         nach HL
FB9B  6F           LD     L,A
FB9C  78           LD     A,B            Länge
FB9D  B1           OR     C              <>0 ?
FB9E  C4 F2 FF     CALL   NZ,FFF2      dann String kopieren
FBA1  D1           POP    DE           Zeiger auf String
FBA2  21 BA B0     LD     HL,B0BA      neuer Zeiger auf Descriptor
FBA5  C9           RET

**********************************   Stringdescriptor kopieren
                                     IN : DE: Zeiger Quelldescriptor
                                          HL: Zeiger Zieldescriptor
FBA6  1A           LD     A,(DE)
FBA7  13           INC    DE             Länge
FBA8  77           LD     (HL),A
FBA9  23           INC    HL
FBAA  1A           LD     A,(DE)
FBAB  13           INC    DE
FBAC  77           LD     (HL),A         und Stringadresse
FBAD  23           INC    HL
FBAE  1A           LD     A,(DE)
FBAF  13           INC    DE             kopieren
FBB0  77           LD     (HL),A
FBB1  23           INC    HL
FBB2  C9           RET

**********************************   Stringdescriptorstack init.
FBB3  21 9C B0     LD     HL,B09C        Startwert
FBB6  22 9A B0     LD     (B09A),HL      als Stringstackpointer
FBB9  C9           RET

**********************************   Descriptor auf Stack u. nach FAC
                                     IN : Descriptor bei $B0BA
FBBA  3E 03        LD     A,03           Typ für String
FBBC  32 C1 B0     LD     (B0C1),A       FAC-Typflag setzen
FBBF  2A 9A B0     LD     HL,(B09A)      Stringstackpointer
FBC2  22 C2 B0     LD     (B0C2),HL      als Zeiger auf Descr. nach FAC
FBC5  11 BA B0     LD     DE,B0BA        obere Stringstackgrenze+1
FBC8  CD B8 FF     CALL   FFB8           mit Stringstackpointer vergl.
FBCB  1E 10        LD     E,10           String expression too complex
FBCD  CA 94 CA     JP     Z,CA94         ggf. Fehler ausgeben
FBD0  11 BA B0     LD     DE,B0BA        Zeiger auf Descriptor
FBD3  CD A6 FB     CALL   FBA6           Descriptor in Stack kopieren
FBD6  22 9A B0     LD     (B09A),HL      neuen Stackpointer setzen
FBD9  C9           RET

**********************************   String aus Stringb./Stack löschen
                                     IN : String im FAC
                                     OUT: DE: Stringadresse
                                          A,B: Stringlänge
                                          Z=1, wenn Leerstring
FBDA  E5           PUSH   HL
FBDB  CD 3C FF     CALL   FF3C           Test auf String im FAC
FBDE  2A C2 B0     LD     HL,(B0C2)      Zeiger auf Descriptor
FBE1  CD E8 FB     CALL   FBE8           ggf. aus Bereich/Stack löschen
FBE4  E1           POP    HL
```

```
FBE5  78           LD    A,B          Stringlänge
FBE6  B7           OR    A
FBE7  C9           RET

***********************************  String aus Stringb./Stack löschen
                                     IN : HL: Zeiger auf Descriptor
                                     OUT: DE: Stringadresse; B: Länge
                                          Z=1, wenn gelöscht
FBE8  CD FF FB     CALL  FBFF           Descr. ggf. vom Stringstack
FBEB  C0           RET   NZ             Descr. nicht gelöscht ?
FBEC  D5           PUSH  DE             Zeiger auf String
FBED  1B           DEC   DE             Zeichen davor
FBEE  2A 8D B0     LD    HL,(B08D)      Zeiger vor Stringbereich
FBF1  CD B8 FF     CALL  FFB8           vergleichen
FBF4  20 07        JR    NZ,FBFD        nicht letzter String im Ber. ?
FBF6  58           LD    E,B            Stringlänge lo
FBF7  16 00        LD    D,00           Länge hi=0
FBF9  19           ADD   HL,DE          addieren
FBFA  22 8D B0     LD    (B08D),HL      als neuen Start der Strings
FBFD  D1           POP   DE             Stringadresse
FBFE  C9           RET

***********************************  Desc. ggf. v. Stringstack löschen
                                     IN : HL: Zeiger auf Descriptor
                                     OUT: HL wie IN
                                          DE: Stringadresse
                                          B: Stringlänge
                                          Z=1, wenn vom Stack gelöscht
FBFF  E5           PUSH  HL             Zeiger auf Descriptor
FC00  46           LD    B,(HL)         Stringlänge
FC01  23           INC   HL
FC02  7E           LD    A,(HL)           und Stringadresse
FC03  23           INC   HL               aus Descriptor laden
FC04  66           LD    H,(HL)
FC05  6F           LD    L,A
FC06  E3           EX    (SP),HL        Adr. retten, Descr.-Zg. zurück
FC07  EB           EX    DE,HL          Descriptor-Zeiger nach DE
FC08  2A 9A B0     LD    HL,(B09A)      Stringdescriptorstackpointer
FC0B  2B           DEC   HL               -3 (Größe eines Eintrags)
FC0C  2B           DEC   HL               = Zeiger auf obersten
FC0D  2B           DEC   HL               Stringstackeintrag
FC0E  CD B8 FF     CALL  FFB8           mit Descr.-Zeiger vergleichen
FC11  20 03        JR    NZ,FC16        ungleich ?
FC13  22 9A B0     LD    (B09A),HL      neuen Stackpointer setzen
FC16  EB           EX    DE,HL          Descriptor-Zeiger nach HL
FC17  D1           POP   DE             Zeiger auf String
FC18  C9           RET

***********************************  Platz für String reservieren
                                     IN : A: benötigte Länge
                                     OUT: Descriptor ab $B0BA
                                          DE: Zeiger auf Platz
FC19  F5           PUSH  AF
FC1A  C5           PUSH  BC
FC1B  E5           PUSH  HL
FC1C  F5           PUSH  AF             Länge
FC1D  CD D1 F5     CALL  F5D1           Platz im Stringber. reserv.
FC20  F1           POP   AF             Länge zurück
```

```
FC21  21 BA B0    LD     HL,B0BA       Zeiger für Descriptor
FC24  77          LD     (HL),A        Länge
FC25  23          INC    HL
FC26  73          LD     (HL),E          und Adresse
FC27  23          INC    HL              eintragen
FC28  72          LD     (HL),D
FC29  E1          POP    HL
FC2A  C1          POP    BC
FC2B  F1          POP    AF
FC2C  C9          RET

*********************************** Basic-Funktion FRE
FC2D  CD 45 FF    CALL   FF45          Typflag des FAC holen
FC30  20 06       JR     NZ,FC38       kein String ?
FC32  CD DA FB    CALL   FBDA          sonst String vom Stringstack
FC35  CD 3E FC    CALL   FC3E          Garbage collection
FC38  CD 28 F6    CALL   F628          Größe des freien Platzes holen
FC3B  C3 60 FE    JP     FE60          in positive REAL-Zahl wandeln

*********************************** Garbage collection
FC3E  C5          PUSH   BC
FC3F  D5          PUSH   DE
FC40  E5          PUSH   HL
FC41  2A 8F B0    LD     HL,(B08F)     Ende des Stringbereichs
FC44  22 8D B0    LD     (B08D),HL     als Anfang setzen
FC47  21 00 00    LD     HL,0000       Flag f. keinen Descr. gefunden
FC4A  22 BD B0    LD     (B0BD),HL     als Zeiger auf Descriptor-Ende
FC4D  2A 89 AE    LD     HL,(AE89)     Zeiger auf Ende der Felder
FC50  22 BF B0    LD     (B0BF),HL     als höchste Stringadresse
FC53  CD 7B FC    CALL   FC7B          hö. Str.-A. außerh. Stringber.
FC56  2A BD B0    LD     HL,(B0BD)     Ende des zugeh. Descriptors
FC59  7C          LD     A,H             keine Stringadr. außerhalb
FC5A  B5          OR     L               Stringbereich gefunden ?
FC5B  28 1A       JR     Z,FC77        d. alle Strings wieder im Ber.
FC5D  56          LD     D,(HL)
FC5E  2B          DEC    HL              Stringadresse aus
FC5F  5E          LD     E,(HL)          Descriptor laden
FC60  E5          PUSH   HL            Zeiger auf Stringadresse
FC61  2B          DEC    HL            Zeiger auf Länge
FC62  4E          LD     C,(HL)        Stringlänge
FC63  06 00       LD     B,00          Länge hi=0
FC65  2A 8D B0    LD     HL,(B08D)     Zeiger vor Stringbereich
FC68  EB          EX     DE,HL         nach DE
FC69  09          ADD    HL,BC         Länge zu Stringadr. addieren
FC6A  2B          DEC    HL            Zeiger auf letztes Stringbyte
FC6B  CD F5 FF    CALL   FFF5          String unter Stringber. kop.
FC6E  13          INC    DE            neuer Zeiger auf String
FC6F  E1          POP    HL            Zeiger auf Stringadresse
FC70  73          LD     (HL),E          neue Stringadresse
FC71  23          INC    HL              in Descriptor eintragen
FC72  72          LD     (HL),D
FC73  1B          DEC    DE            Zeiger vor String
FC74  EB          EX     DE,HL         als neuen Start der Strings
FC75  18 CD       JR     FC44          nächsten String in Stringber.
FC77  E1          POP    HL
FC78  D1          POP    DE
FC79  C1          POP    BC
FC7A  C9          RET
```

```
***********************************   höch. Stringadr. außerh. Ber. su.
FC7B  21 9C B0     LD     HL,B09C        Zeiger auf Stringstack
FC7E  ED 5B 9A B0  LD     DE,(B09A)      Stringstackpointer
FC82  CD B8 FF     CALL   FFB8           Stringstackende erreicht ?
FC85  28 0F        JR     Z,FC96         dann Variablen durchgehen
FC87  7E           LD     A,(HL)         Länge
FC88  23           INC    HL
FC89  4E           LD     C,(HL)           und Adresse aus
FC8A  23           INC    HL               Descriptor laden
FC8B  46           LD     B,(HL)
FC8C  E5           PUSH   HL             Zeiger in Stringstack
FC8D  EB           EX     DE,HL          als Descriptorende nach DE
FC8E  B7           OR     A              Länge <>0 ?
FC8F  C4 9C FC     CALL   NZ,FC9C        dann ggf. als höchste Adresse
FC92  E1           POP    HL             Zeiger in Stringstack
FC93  23           INC    HL             Zeiger auf nächsten Descriptor
FC94  18 E8        JR     FC7E           Stringstack weiter durchgehen

FC96  11 9C FC     LD     DE,FC9C        Adresse der Routine
FC99  C3 74 DA     JP     DA74           sämtliche Stringv. durchgehen

***********************************   ggf. höchste Stringadresse setzen
                                      IN : BC: Stringadresse
                                           DE: Zeiger auf Descr.-Ende
FC9C  2A 8D B0     LD     HL,(B08D)      Zeiger auf Start der Strings
FC9F  CD BE FF     CALL   FFBE           mit Stringadresse vergleichen
FCA2  D8           RET    C              String schon im Stringber. ?
FCA3  2A BF B0     LD     HL,(B0BF)      bisherige höchste Stringadr.
FCA6  CD BE FF     CALL   FFBE           mit neuer Adresse vergleichen
FCA9  D0           RET    NC             neue Adresse kleiner ?
FCAA  EB           EX     DE,HL          Zeiger auf Descriptor-Ende
FCAB  22 BD B0     LD     (B0BD),HL      setzen
FCAE  ED 43 BF B0  LD     (B0BF),BC      Adr. als höchste Stringadresse
FCB2  C9           RET

***********************************   Parameter für Dezimalwandl. holen
                                      IN : Zahl im FAC
                                      OUT: HL: Zeiger auf höchstwert.
                                               Byte der Zahl
                                           B: Vorzeichen
                                           C: Zahl der signifik. Bytes
                                           E: Kommaposition
FCB3  CD 2D FF     CALL   FF2D           Test auf numerisch, Typ nach C
FCB6  D2 52 BD     JP     NC,BD52        REAL-Zahl ?
FCB9  CD A3 BD     CALL   BDA3           Parameter f. Integerzahl holen
FCBC  22 C2 B0     LD     (B0C2),HL      Zahl in FAC speichern
FCBF  21 C3 B0     LD     HL,B0C3        Zeiger auf höchstwertiges Byte
FCC2  C9           RET

***********************************   Wand.-Param. f. pos. Integer hol.
                                      IN : Zahl im FAC
                                      OUT: HL: Zeiger auf höchstwert. Byte der Zahl
                                           B: Vorzeichen
                                           C: Zahl der signifik. Bytes
                                           E: Kommaposition
FCC3  CD C2 FE     CALL   FEC2           UNT-Funktion, FAC nach Integer
FCC6  21 C3 B0     LD     HL,B0C3        Zeiger auf höchstwertiges Byte
FCC9  C3 A6 BD     JP     BDA6           Parameter f. pos. Integer hol.
```

```
**********************************    Basic-Operator +
                                      IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden; 2. Operand im FAC
                                      OUT: Ergebnis im FAC
FCCC  CD 15 FE     CALL   FE15           Typen der Operanden angleichen
FCCF  30 09        JR     NC,FCDA        REAL-Operanden ?
FCD1  CD AC BD     CALL   BDAC           sonst Integeraddition
FCD4  DA 0D FF     JP     C,FF0D         kein Fehler ? dann Erg. n. FAC
FCD7  CD 4F FE     CALL   FE4F           Operanden nach REAL wandeln
FCDA  CD 58 BD     CALL   BD58           REAL-Addition
FCDD  D8           RET    C              kein Fehler ?
FCDE  C3 F3 CA     JP     CAF3           sonst "Overflow" ausgeben

**********************************    Basic-Operator -
                                      IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden; 2. Operand im FAC
                                      OUT: Ergebnis im FAC
FCE1  CD 15 FE     CALL   FE15           Typen der Operanden angleichen
FCE4  30 09        JR     NC,FCEF        REAL-Operanden ?
FCE6  CD B2 BD     CALL   BDB2           sonst Integersubtraktion
FCE9  DA 0D FF     JP     C,FF0D         kein Fehler ? dann Erg. n. FAC
FCEC  CD 4F FE     CALL   FE4F           Operanden nach REAL wandeln
FCEF  CD 5E BD     CALL   BD5E           REAL-Subtraktion
FCF2  D8           RET    C              kein Fehler ?
FCF3  18 E9        JR     FCDE           sonst "Overflow" ausgeben

**********************************    Basic-Operator *
                                      IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden; 2. Operand im FAC
                                      OUT: Ergebnis im FAC
FCF5  CD 15 FE     CALL   FE15           Typen der Operanden angleichen
FCF8  30 09        JR     NC,FD03        REAL-Operanden ?
FCFA  CD B5 BD     CALL   BDB5           sonst Integermultiplikation
FCFD  DA 0D FF     JP     C,FF0D         kein Fehler ? dann Erg. n. FAC
FD00  CD 4F FE     CALL   FE4F           Operanden nach REAL wandeln
FD03  CD 61 BD     CALL   BD61           REAL-Multiplikation
FD06  D8           RET    C              kein Fehler ?
FD07  18 D5        JR     FCDE           sonst "Overflow" ausgeben

**********************************    numerischer Vergleich
                                      IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden; 2. Operand im FAC
                                      OUT: Ergebnis im FAC
                                           A: Vergleichsergebnis
                                            $00 f. gleich
                                            $FF f. 1. Operand größer
                                            $01 f. 1. Operand kleiner
FD09  CD 15 FE     CALL   FE15           Typen der Operanden angleichen
FD0C  DA C4 BD     JP     C,BDC4         Integer ? d. Integervergleich
FD0F  C3 6A BD     JP     BD6A           sonst REAL-Vergleich

**********************************    Basic-Operator /
                                      IN : HL: Zeiger auf 1. Operanden
                                           1. Oper. oberh. Basic-Stack
                                           C: Typ des 1. Operanden
                                           2. Operand im FAC
                                      OUT: Ergebnis im FAC
FD12  3A C1 B0     LD     A,(B0C1)       Typflag des FAC (2. Operanden)
```

```
FD15  B1          OR    C           Typflag des 1. Operanden
FD16  FE 02       CP    02          nicht beide Integer ?
FD18  20 05       JR    NZ,FD1F     dann Typen angleichen
FD1A  CD 4F FE    CALL  FE4F        Operanden nach REAL wandeln
FD1D  18 03       JR    FD22
FD1F  CD 15 FE    CALL  FE15        Typen angleichen (auf REAL!)
FD22  EB          EX    DE,HL       Operanden vertauschen
FD23  D5          PUSH  DE          Zeiger auf 2. Operanden im FAC
FD24  CD 64 BD    CALL  BD64        REAL-Division
FD27  D1          POP   DE          Zeiger auf FAC
FD28  F5          PUSH  AF          Fehlerflags retten
FD29  01 05 00    LD    BC,0005     Größe des FAC
FD2C  CD F2 FF    CALL  FFF2        Ergebnis nach FAC kopieren
FD2F  F1          POP   AF          Fehlerflags
FD30  D8          RET   C           kein Fehler ?
FD31  CA EA CA    JP    Z,CAEA      ggf. "Division by zero"
FD34  C3 F3 CA    JP    CAF3        "Overflow" ausgeben

***********************************  Basic-Operator \
                                     IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden
                                           2. Operand im FAC
                                     OUT: Ergebnis im FAC
FD37  CD 9A FE    CALL  FE9A            Operanden nach Integer
FD3A  EB          EX    DE,HL           Operanden vertauschen
FD3B  CD B8 BD    CALL  BDB8            Integerdivision
FD3E  DA 0D FF    JP    C,FF0D          kein Fehler ? d. Erg. nach FAC
FD41  28 10       JR    Z,FD53          ggf. "Division by zero"
FD43  21 00 80    LD    HL,8000         Überlauf ? dann 32768
FD46  C3 60 FE    JP    FE60            in positive REAL-Zahl wandeln

***********************************  Basic-Operator MOD
                                     IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden
                                           2. Operand im FAC
                                     OUT: Ergebnis im FAC
FD49  CD 9A FE    CALL  FE9A            Operanden nach Integer
FD4C  EB          EX    DE,HL           Operanden vertauschen
FD4D  CD BB BD    CALL  BDBB            Integer-Divisionrest berechnen
FD50  DA 0D FF    JP    C,FF0D          kein Fehler ? d. Erg. nach FAC
FD53  1E 0B       LD    E,0B            Nr. für "Division by zero"
FD55  C3 94 CA    JP    CA94            Fehler ausgeben

***********************************  Basic-Operator AND
                                     IN : HL: Zeiger auf 1. Operanden
                                           C: Typ des 1. Operanden
                                           2. Operand im FAC
                                     OUT: Ergebnis im FAC
FD58  CD 9A FE    CALL  FE9A            Operanden nach Integer
FD5B  7B          LD    A,E
FD5C  A5          AND   L                  AND-Verknüpfung
FD5D  6F          LD    L,A
FD5E  7C          LD    A,H
FD5F  A2          AND   D
FD60  C3 0C FF    JP    FF0C            Ergebnis nach FAC
```

```
*********************************   Basic-Operator OR
                                    IN : HL: Zeiger auf 1. Operanden
                                         C: Typ des 1. Operanden
                                         2. Operand im FAC
                                    OUT: Ergebnis im FAC
FD63  CD 9A FE    CALL  FE9A           Operanden nach Integer
FD66  7B          LD    A,E
FD67  B5          OR    L                 OR-Verknüpfung
FD68  6F          LD    L,A
FD69  7A          LD    A,D
FD6A  B4          OR    H
FD6B  18 F3       JR    FD60           Ergebnis nach FAC

*********************************   Basic-Operator XOR
                                    IN : HL: Zeiger auf 1. Operanden
                                         C: Typ des 1. Operanden
                                         2. Operand im FAC
                                    OUT: Ergebnis im FAC
FD6D  CD 9A FE    CALL  FE9A           Operanden nach Integer
FD70  7B          LD    A,E
FD71  AD          XOR   L                 XOR-Verknüpfung
FD72  6F          LD    L,A
FD73  7C          LD    A,H
FD74  AA          XOR   D
FD75  18 E9       JR    FD60           Ergebnis nach FAC

*********************************   Basic-Operator NOT
FD77  E5          PUSH  HL             Basic-PC
FD78  CD 8D FE    CALL  FE8D           CINT, FAC nach Integer nach HL
FD7B  7D          LD    A,L
FD7C  2F          CPL                     HL komplementieren
FD7D  6F          LD    L,A
FD7E  7C          LD    A,H
FD7F  2F          CPL
FD80  CD 0C FF    CALL  FF0C           Ergebnis nach FAC
FD83  E1          POP   HL             Basic-PC
FD84  C9          RET

*********************************   Basic-Funktion ABS
                                    IN/OUT: Zahl im FAC
                                    OUT: HL: Integerzahl bzw.
                                             Zeiger auf REAL-Zahl
FD85  CD A3 FD    CALL  FDA3           Vorzeichen von FAC holen
FD88  F0          RET   P              positiv ? dann fertig
FD89  E5          PUSH  HL             Zeiger bzw. Zahl retten
FD8A  C5          PUSH  BC
FD8B  CD 2D FF    CALL  FF2D           Typflag und Argument holen
FD8E  30 0D       JR    NC,FD9D        REAL-Zahl ?
FD90  CD C7 BD    CALL  BDC7           Integer-Vorzeichenwechsel
FD93  22 C2 B0    LD    (B0C2),HL      Integerwert wieder speichern
FD96  D5          PUSH  DE
FD97  D4 60 FE    CALL  NC,FE60        Überlauf ? dann nach pos. REAL
FD9A  D1          POP   DE
FD9B  18 03       JR    FDA0
FD9D  CD 6D BD    CALL  BD6D           REAL-Vorzeichenwechsel
FDA0  C1          POP   BC
FDA1  E1          POP   HL             Zahl bzw. Zeiger auf Zahl
FDA2  C9          RET
```

```
***********************************   Vorzeichen von FAC holen
                                      IN : Zahl im FAC
                                      OUT: A: Vorzeichen
                                           $01, S=0 für positiv
                                           $00, S=0 für Zahl=0
                                           $FF, S=1 für negativ
FDA3  CD 2D FF     CALL   FF2D          Typflag und Wert holen
FDA6  DA CA BD     JP     C,BDCA        Integer ?
FDA9  C5           PUSH   BC
FDAA  CD 70 BD     CALL   BD70          REAL-Vorzeichen holen
FDAD  C1           POP    BC
FDAE  C9           RET

***********************************   Zahl runden, nach FAC
                                      IN : HL: Zeiger auf Zahl
                                            C: Typ der Zahl
                                            B: Rundungsexponent
FDAF  E5           PUSH   HL            Zeiger auf Zahl
FDB0  79           LD     A,C           Typ der Zahl
FDB1  CD 4B FF     CALL   FF4B          Zahl in FAC kopieren
FDB4  D1           POP    DE            Zeiger auf Zahl
FDB5  CD 2D FF     CALL   FF2D          Typ und Wert holen
FDB8  78           LD     A,B           Rundungsexponent
FDB9  30 0B        JR     NC,FDC6       REAL-Zahl ?
FDBB  B7           OR     A             Rundungsexponent positiv ?
FDBC  F0           RET    P             dann fertig, da Integer-Zahl
FDBD  CD 6A FE     CALL   FE6A          Zahl nach REAL
FDC0  CD CE FD     CALL   FDCE          REAL-Wert runden
FDC3  C3 8D FE     JP     FE8D          CINT, FAC wieder nach Integer
FDC6  B7           OR     A             Rundungsexponent <>0 ?
FDC7  20 05        JR     NZ,FDCE       dann entsprechend runden
FDC9  11 49 BD     LD     DE,BD49       Routine REAL n. 4-Byte-Integer
FDCC  18 26        JR     FDF4          FAC a. letzte Vork.-St. runden

FDCE  D5           PUSH   DE            Zeiger auf ursprüngliche Zahl
FDCF  C5           PUSH   BC            Rundungsexponent retten
FDD0  78           LD     A,B           Rundungsexponent
FDD1  CD 55 BD     CALL   BD55          Zahl m. 10^R.-Exp. multipliz.
FDD4  DC 49 BD     CALL   C,BD49        o.k. ? d. nach 4-Byte-Integer
FDD7  78           LD     A,B           Vorzeichen
FDD8  C1           POP    BC            Rundungsexponent
FDD9  D1           POP    DE            Zeiger auf ursprüngliche Zahl
FDDA  30 08        JR     NC,FDE4       Fehler b. Mult. ? d. urspr. Z.
FDDC  CD 43 BD     CALL   BD43          4-Byte-Integer wieder n. REAL
FDDF  AF           XOR    A             Null
FDE0  90           SUB    B             - Rundungsexponent
FDE1  C3 55 BD     JP     BD55          Zahl durch 10^R.-Exp. teilen

FDE4  EB           EX     DE,HL         Zeiger auf ursprüngliche Zahl
FDE5  C3 4E FF     JP     FF4E          Zahl wieder nach FAC

***********************************   Basic-Funktion FIX
FDE8  11 4C BD     LD     DE,BD4C       Adresse für FLO FIX
FDEB  18 03        JR     FDF0          Funktion ausführen
```

```
**********************************  Basic-Funktion INT
FDED  11 4F BD    LD    DE,BD4F      Adresse für FLO INT
FDF0  CD 2D FF    CALL  FF2D         Typflag und Wert holen
FDF3  D8          RET   C            Integer ? dann fertig
FDF4  CD FB FF    CALL  FFFB         Routine ausführen
FDF7  D0          RET   NC           Zahl zu groß f. Nachkommast. ?
FDF8  3A C1 B0    LD    A,(B0C1)     Typ des FAC (??)
FDFB  CD 06 FE    CALL  FE06         Zahl nach Integer
FDFE  D8          RET   C            nicht zu groß für Integer ?
FDFF  CD 1D FF    CALL  FF1D         Typ und Zeiger auf FAC
FE02  78          LD    A,B          Vorzeichen
FE03  C3 43 BD    JP    BD43         4-Byte-Integer nach REAL

**********************************  Integer m. Vorzeichen n. Integer
                                    IN : HL: Zeiger auf Integerzahl
                                          C: Länge der Integerzahl
                                          B: Vorzeichen
                                    OUT: CY=0 bei zu groß f. Integer
FE06  79          LD    A,C          Länge der Integerzahl
FE07  FE 03       CP    03           zu viele Bytes f. Integer ?
FE09  D0          RET   NC           dann zurück
FE0A  7E          LD    A,(HL)         Integerzahl
FE0B  23          INC   HL             laden
FE0C  66          LD    H,(HL)
FE0D  6F          LD    L,A
FE0E  CD A9 BD    CALL  BDA9         Vorzeichen setzen ?
FE11  D0          RET   NC           Zahl zu groß ?
FE12  C3 0D FF    JP    FF0D         sonst Integer nach FAC

**********************************  Typen angleichen, Werte holen
                                    IN : HL: Zeiger auf 1. Operanden
                                          C: Typ des Operanden
                                          2. Operand im FAC
                                    OUT: CY=0 für REAL
                                          DE: Zeiger auf 1. Operanden
                                          HL: Zeiger auf 2. Operanden
                                          CY=1 für Integer
                                          DE: 1. Operand
                                          HL: 2. Operand
FE15  79          LD    A,C          Typ des 1. Operanden
FE16  FE 03       CP    03           String ?
FE18  28 32       JR    Z,FE4C       dann Fehler
FE1A  3A C1 B0    LD    A,(B0C1)     Typ des 2. Operanden
FE1D  FE 03       CP    03           String ?
FE1F  28 2B       JR    Z,FE4C       dann Fehler
FE21  B9          CP    C            Typen vergleichen ?
FE22  28 17       JR    Z,FE3B       gleich ? dann Werte/Zeiger h.
FE24  30 0C       JR    NC,FE32      2. Operand REAL, 1. Integer ?
FE26  E5          PUSH  HL           Zeiger auf 1. Operand
FE27  21 C1 B0    LD    HL,B0C1      Zeiger auf Typflag des FAC
FE2A  71          LD    (HL),C       Typflag auf REAL setzen
FE2B  23          INC   HL           Zeiger auf FAC (2. Operanden)
FE2C  CD 63 FE    CALL  FE63         nach REAL wandeln
FE2F  D1          POP   DE           Zeiger auf 1. Operanden
FE30  B7          OR    A            CY=0 für REAL
FE31  C9          RET
FE32  CD 63 FE    CALL  FE63         1. Operand nach REAL wandeln
FE35  EB          EX    DE,HL        Zeiger auf 1. Operanden n. DE
```

```
FE36  21 C2 B0    LD    HL,B0C2       Zeiger auf 2. Operanden (FAC)
FE39  B7          OR    A             CY=0 für REAL
FE3A  C9          RET
FE3B  EE 02       XOR   02            beide Integer ?
FE3D  28 05       JR    Z,FE44        dann laden
FE3F  EB          EX    DE,HL         Zeiger auf 1. Operanden n. DE
FE40  21 C2 B0    LD    HL,B0C2       Zeiger auf 2. Operanden (FAC)
FE43  C9          RET
FE44  5E          LD    E,(HL)        1. Operanden
FE45  23          INC   HL            laden, nach DE
FE46  56          LD    D,(HL)
FE47  2A C2 B0    LD    HL,(B0C2)     2. Operanden aus FAC nach HL
FE4A  37          SCF                 CY=1 für Integer
FE4B  C9          RET

FE4C  C3 40 FF    JP    FF40          "Type mismatch"

*********************************** Integeroperanden n. REAL wandeln
                                    IN : 1. Operand auf Basic-Stack
                                         2. Operand im FAC
                                    OUT: DE: Zeiger auf 1. Operanden
                                         HL: Zeiger auf 2. Operanden
FE4F  2A C2 B0    LD    HL,(B0C2)     2. Operanden aus FAC laden
FE52  CD 6A FE    CALL  FE6A          nach REAL in FAC wandeln
FE55  2A 8B B0    LD    HL,(B08B)     Zg. auf Integer in Basic-Stack
FE58  CD 63 FE    CALL  FE63          nach REAL wandeln
FE5B  EB          EX    DE,HL         Adresse d. 1. Operanden n. DE
FE5C  21 C2 B0    LD    HL,B0C2       Adresse des 2. Operanden (FAC)
FE5F  C9          RET

*********************************** positive Integerzahl nach REAL
                                    IN : HL: Integerzahl
                                    OUT: REAL-Zahl im FAC
                                         DE: Zeiger auf REAL-Zahl
FE60  AF          XOR   A             positives Vorzeichen
FE61  18 08       JR    FE6B

*********************************** Integer nach REAL wandeln
                                    IN : HL: Zeiger auf Integerzahl
                                             und Platz für REAL-Zahl
                                    OUT: HL: Zeiger auf REAL-Zahl
FE63  5E          LD    E,(HL)
FE64  23          INC   HL              Integerwert laden, nach DE
FE65  56          LD    D,(HL)
FE66  2B          DEC   HL            Zeiger als Platz für REAL-Zahl
FE67  7A          LD    A,D           Hi-Byte als Vorzeichen
FE68  18 08       JR    FE72

*********************************** Integer nach REAL wandeln
                                    IN : HL: Integerzahl
                                    OUT: REAL-Zahl im FAC
                                         HL: Zeiger auf REAL-Zahl
FE6A  7C          LD    A,H           Hi-Byte als Vorzeichen
FE6B  EB          EX    DE,HL         Integerwert nach DE
FE6C  21 C1 B0    LD    HL,B0C1       Zeiger auf Typflag des FAC
FE6F  36 05       LD    (HL),05       Typ für REAL setzen
FE71  23          INC   HL            Zeiger auf FAC
FE72  EB          EX    DE,HL         Zeiger nach DE, Wert nach HL
```

```
FE73  F5          PUSH   AF          Vorzeichen
FE74  B7          OR     A           Vorzeichen negativ ?
FE75  FC C7 BD    CALL   M,BDC7      dann invertieren (Betrag ber.)
FE78  F1          POP    AF          Vorzeichen
FE79  C3 40 BD    JP     BD40        Integer m. Vorz. nach REAL

***********************************  4-Byte-Integer nach REAL
                                     IN : HL: Lo-Word
                                          DE: Hi-Word
FE7C  22 C2 B0    LD     (B0C2),HL   Lo-Word
FE7F  EB          EX     DE,HL
FE80  22 C4 B0    LD     (B0C4),HL   und Hi-Word in FAC speichern
FE83  21 C1 B0    LD     HL,B0C1     Zeiger auf Typflag des FAC
FE86  36 05       LD     (HL),05     Typflag auf REAL
FE88  23          INC    HL          Zeiger auf FAC
FE89  AF          XOR    A           Vorzeichen positiv
FE8A  C3 43 BD    JP     BD43        4-Byte-Integer nach REAL

***********************************  Basic-Funktion CINT
FE8D  CD 93 FE    CALL   FE93        REAL in FAC nach Integer
FE90  D8          RET    C           kein Fehler ?
FE91  18 3F       JR     FED2        sonst "Overflow"

***********************************  REAL im FAC nach Integer im FAC
FE93  CD A5 FE    CALL   FEA5        REAL im FAC nach Integer in HL
FE96  22 C2 B0    LD     (B0C2),HL   Integer in FAC speichern
FE99  C9          RET

***********************************  Operanden nach Integer wandeln
                                     IN : HL: Zeiger auf 1. Operanden
                                          C: Typ des 1. Operanden; 2. Operand im FAC
                                     OUT: DE: 1. Operand
                                          HL/FAC: 2. Operand
FE9A  79          LD     A,C         Typ des 1. Operanden
FE9B  CD AC FE    CALL   FEAC        1. Operanden nach Integer
FE9E  EB          EX     DE,HL       Operanden vertauschen
FE9F  DC A5 FE    CALL   C,FEA5      ggf. 2. Operanden nach Integer
FEA2  D8          RET    C           kein Fehler ?
FEA3  18 2D       JR     FED2        sonst "Overflow"

***********************************  FAC nach Integer
                                     IN : Zahl im FAC
                                     OUT: HL: Integerzahl
                                     CY=0 für Überlauf
FEA5  21 C1 B0    LD     HL,B0C1     Zeiger auf Typ des FAC
FEA8  7E          LD     A,(HL)      Typ des FAC
FEA9  36 02       LD     (HL),02     neuer Typ = Integer
FEAB  23          INC    HL          Zeiger auf FAC
FEAC  FE 03       CP     03
FEAE  38 0D       JR     C,FEBD      Integer im FAC ? dann laden
FEB0  CA 40 FF    JP     Z,FF40      String ? dann "Type mismatch"
FEB3  C5          PUSH   BC
FEB4  CD 46 BD    CALL   BD46        REAL nach Integer m. Vorz.
FEB7  47          LD     B,A         Vorzeichen
FEB8  DC A9 BD    CALL   C,BDA9      Integer m. Vorz. nach Integer
FEBB  C1          POP    BC
FEBC  C9          RET
FEBD  7E          LD     A,(HL)        Integerzahl
```

```
FEBE  23           INC    HL             aus FAC laden,
FEBF  66           LD     H,(HL)         nach HL
FEC0  6F           LD     L,A
FEC1  C9           RET

**********************************    Basic-Funktion UNT
FEC2  CD 2D FF     CALL   FF2D            Typflag und Wert holen
FEC5  D8           RET    C               Integer ?
FEC6  CD 46 BD     CALL   BD46            REAL nach Integer m. Vorz.
FEC9  30 07        JR     NC,FED2         Fehler ? dann "Overflow"
FECB  47           LD     B,A             Vorzeichen negativ ?
FECC  FC A9 BD     CALL   M,BDA9          d. Integer m. Vorz. n. Integer
FECF  DA 0D FF     JP     C,FF0D          kein Fehler ? d. Erg. nach FAC
FED2  1E 06        LD     E,06            Nr. für "Overflow"
FED4  C3 94 CA     JP     CA94            Fehler ausgeben

**********************************    FAC-Typ angleichen
                                      IN : A: gewünschter Typ
FED7  E5           PUSH   HL
FED8  D5           PUSH   DE
FED9  C5           PUSH   BC
FEDA  21 C1 B0     LD     HL,B0C1         Zeiger auf FAC-Typ
FEDD  BE           CP     (HL)            Typen nicht gleich ?
FEDE  C4 E5 FE     CALL   NZ,FEE5         dann FAC angleichen
FEE1  C1           POP    BC
FEE2  D1           POP    DE
FEE3  E1           POP    HL
FEE4  C9           RET

**********************************    FAC-Typ angleichen
                                      IN : A: gewünschter Typ
FEE5  D6 03        SUB    03
FEE7  38 A4        JR     C,FE8D          Integer gew. ? dann CINT
FEE9  CA 3C FF     JP     Z,FF3C          String gew. ? d. Test auf Str.

**********************************    Basic-Funktion CREAL
                                      OUT: HL: Zeiger auf REAL-Zahl
FEEC  CD 2D FF     CALL   FF2D            Typ und Wert/Zeiger holen
FEEF  DA 6A FE     JP     C,FE6A          Integer ? dann nach REAL
FEF2  C9           RET

**********************************    FAC löschen (FAC=0)
FEF3  E5           PUSH   HL
FEF4  21 00 00     LD     HL,0000
FEF7  22 C2 B0     LD     (B0C2),HL
FEFA  22 C4 B0     LD     (B0C4),HL
FEFD  22 C5 B0     LD     (B0C5),HL
FF00  E1           POP    HL
FF01  C9           RET

**********************************    Basic-Funktion SGN
FF02  CD A3 FD     CALL   FDA3            Vorzeichen von FAC holen

**********************************    Byte in A nach Integer in FAC
FF05  6F           LD     L,A             Byte ins Lo-Byte
FF06  87           ADD    A               Vorzeichen ins Carry
FF07  9F           SBC    A               Vorzeichen erweitern (0/$FF)
FF08  18 02        JR     FF0C            und ins Hi-Byte
```

```
*********************************** pos. Byte in A n. Integer in FAC
FF0A  6F          LD    L,A          Lo-Byte
FF0B  AF          XOR   A            Null
FF0C  67          LD    H,A          als Hi-Byte setzen

*********************************** Integer in HL nach FAC
FF0D  22 C2 B0    LD    (B0C2),HL    Integerwert in FAC speichern
FF10  3E 02       LD    A,02         Typ für Integer
FF12  32 C1 B0    LD    (B0C1),A     als FAC-Typ setzen
FF15  C9          RET

*********************************** FAC auf REAL, Zeiger nach HL
FF16  21 C2 B0    LD    HL,B0C2      Zeiger auf FAC
FF19  3E 05       LD    A,05         Typ für REAL
FF1B  18 F5       JR    FF12         als FAC-Typ setzen

*********************************** Zeiger auf FAC und Typ holen
                                   OUT: HL: Zeiger auf FAC
                                         C: Typ des FAC
FF1D  21 C1 B0    LD    HL,B0C1      Zeiger auf Typ
FF20  4E          LD    C,(HL)       Typ laden
FF21  23          INC   HL           Zeiger auf FAC
FF22  C9          RET

*********************************** Typ des FAC nach A holen
FF23  3A C1 B0    LD    A,(B0C1)
FF26  C9          RET

*********************************** Typ des FAC holen, Flags setzen
                                   OUT: A: Typ des FAC
                                         CY=1, Z=0, wenn Integer
                                         CY=0, Z=1, wenn String
                                         CY=0, Z=0, wenn REAL
FF27  3A C1 B0    LD    A,(B0C1)
FF2A  FE 03       CP    03
FF2C  C9          RET

*********************************** numerischen Wert aus FAC holen
                                   OUT: CY=1, wenn Intger
                                         HL: Integerwert aus FAC
                                         CY=0, wenn REAL
                                         HL: Zeiger auf REAL im FAC
FF2D  3A C1 B0    LD    A,(B0C1)     Typ des FAC
FF30  FE 03       CP    03           String ?
FF32  28 0C       JR    Z,FF40       dann "Type mismatch"
FF34  2A C2 B0    LD    HL,(B0C2)    FAC-Wert laden
FF37  D8          RET   C            Integer ?
FF38  21 C2 B0    LD    HL,B0C2      sonst Zeiger auf FAC
FF3B  C9          RET

*********************************** Test auf String, sonst Fehler
FF3C  CD 45 FF    CALL  FF45         Typ holen, Flags setzen
FF3F  C8          RET   Z            String ?

FF40  1E 0D       LD    E,0D         Nr. für "Type mismatch"
FF42  C3 94 CA    JP    CA94         Fehler ausgeben
```

```
***********************************  Typ des FAC holen, Flags setzen
                                     OUT: A: Typ des FAC
                                          CY=1, Z=0, wenn Integer
                                          CY=0, Z=1, wenn String
                                         CY=0, Z=0, wenn REAL
FF45  3A C1 B0     LD     A,(B0C1)
FF48  FE 03        CP     03
FF4A  C9           RET

***********************************  Wert nach FAC kopieren
                                     IN : HL: Zeiger auf Wert
                                          A: Typ
                                     OUT: HL: Zeiger nach Wert
FF4B  32 C1 B0     LD     (B0C1),A      Typ setzen
FF4E  11 C2 B0     LD     DE,B0C2       Zeiger auf FAC
FF51  18 13        JR     FF66          Wert nach FAC kopieren

***********************************  FAC auf Basic-Stack
                                     OUT: C: Typ des FAC
FF53  D5           PUSH   DE
FF54  E5           PUSH   HL
FF55  3A C1 B0     LD     A,(B0C1)      Typ des FAC als Zahl der Bytes
FF58  4F           LD     C,A           Typ nach C
FF59  CD B0 F5     CALL   F5B0          Platz auf Basic-Stack reserv.
FF5C  CD 62 FF     CALL   FF62          FAC dorthin kopieren
FF5F  E1           POP    HL
FF60  D1           POP    DE
FF61  C9           RET

***********************************  FAC kopieren
                                     IN : HL: Zieladresse
FF62  EB           EX     DE,HL         Zieladresse nach DE
FF63  21 C2 B0     LD     HL,B0C2       Zeiger auf FAC
FF66  C5           PUSH   BC
FF67  3A C1 B0     LD     A,(B0C1)      Zahl der Bytes im FAC
FF6A  4F           LD     C,A           als Länge lo
FF6B  06 00        LD     B,00          Länge hi=0
FF6D  ED B0        LDIR                 Wert kopieren
FF6F  C1           POP    BC
FF70  C9           RET

***********************************  Test auf Buchstabe
                                     IN : A: Zeichen
                                          CY=1, wenn Buchstabe
FF71  CD 8A FF     CALL   FF8A          auf Großschrift forcieren
FF74  FE 41        CP     41            kleiner "A" ?
FF76  3F           CCF
FF77  D0           RET    NC            dann kein Buchstabe
FF78  FE 5B        CP     5B            >= "Z"+1 ?
FF7A  C9           RET

***********************************  Test auf Buchstabe, Ziffer, "."
                                     IN : A: Zeichen
                                     OUT: CY=1 f. Buchst., Ziffer, "."
FF7B  CD 71 FF     CALL   FF71          Test auf Buchstabe
FF7E  D8           RET    C             Buchstabe ?
```

```
*********************************** Test auf Ziffer oder Dezimalpunkt
                                     IN : A: Zeichen
                                     OUT: CY=1 f. Ziffer oder "."
FF7F  FE 2E      CP     2E              "." ?
FF81  37         SCF
FF82  C8         RET    Z               dann CY=1, zurück

*********************************** Test auf Ziffer
                                     IN : A: Zeichen
                                     OUT: CY=1 f. Ziffer
FF83  FE 30      CP     30              kleiner "0" ?
FF85  3F         CCF
FF86  D0         RET    NC              dann keine Ziffer
FF87  FE 3A      CP     3A              >= "9"+1 ?
FF89  C9         RET

*********************************** auf Großschrift forcieren
                                     IN/OUT: A: Zeichen
FF8A  FE 61      CP     61              kleiner "a" ?
FF8C  D8         RET    C               dann zurück
FF8D  FE 7B      CP     7B              >= "z"+1 ?
FF8F  D0         RET    NC              dann zurück
FF90  D6 20      SUB    20              nach Großschrift wandeln
FF92  C9         RET

*********************************** Adr. aus Tabelle entsp. Zeichen
                                     IN : HL: Zeiger auf Tabelle
                                          A: Zeichen
                                     OUT: HL: Adresse
FF93  F5         PUSH   AF
FF94  C5         PUSH   BC
FF95  46         LD     B,(HL)          Zahl der Tabelleneinträge
FF96  23         INC    HL              Zeiger auf 1. Adr. (Default)
FF97  E5         PUSH   HL              retten
FF98  23         INC    HL                Adresse
FF99  23         INC    HL                übergehen
FF9A  BE         CP     (HL)            ges. Zeichen gefunden ?
FF9B  23         INC    HL              Zeiger auf zugehörige Adresse
FF9C  28 04      JR     Z,FFA2          gefunden ? dann Adresse laden
FF9E  05         DEC    B               Zähler für Tabelleneinträge
FF9F  20 F7      JR     NZ,FF98         weitere Einträge ?
FFA1  E3         EX     (SP),HL         Zeiger auf Default-Adr. zurück
FFA2  F1         POP    AF              Adresse vom Stack löschen
FFA3  7E         LD     A,(HL)            Adresse entspr. Zeichen
FFA4  23         INC    HL                bzw. Default-Adresse
FFA5  66         LD     H,(HL)            aus Tabelle laden, nach HL
FFA6  6F         LD     L,A
FFA7  C1         POP    BC
FFA8  F1         POP    AF
FFA9  C9         RET

*********************************** Byte in Tabelle suchen
                                     IN : HL: Zeiger auf Tabelle
                                          A: gesuchtes Byte
                                     OUT: CY=1, wenn gefunden
                                          dann:
                                          HL: Zeiger nach Byte in Tab.
FFAA  C5         PUSH   BC
```

```
FFAB  4F          LD    C,A          gesuchtes Byte
FFAC  7E          LD    A,(HL)       Byte aus Tabelle
FFAD  B7          OR    A            Tabellenende ?
FFAE  28 05       JR    Z,FFB5       dann nicht gefunden
FFB0  23          INC   HL
FFB1  B9          CP    C            Byte=gesuchtes Byte ?
FFB2  20 F8       JR    NZ,FFAC      nein ? dann weiter suchen
FFB4  37          SCF                CY=1 für gefunden
FFB5  79          LD    A,C          Suchbyte wieder nach A
FFB6  C1          POP   BC
FFB7  C9          RET

*********************************** HL und DE vergleichen
                                    OUT: CY=0, Z=1, wenn gleich
                                         CY=0, Z=0, wenn HL größer
                                         CY=1, Z=0, wenn DE größer
FFB8  7C          LD    A,H
FFB9  92          SUB   D
FFBA  C0          RET   NZ
FFBB  7D          LD    A,L
FFBC  93          SUB   E
FFBD  C9          RET

*********************************** HL und BC vergleichen
                                    OUT: CY=0, Z=1, wenn gleich
                                         CY=0, Z=0, wenn HL größer
                                         CY=1, Z=0, wenn BC größer
FFBE  7C          LD    A,H
FFBF  90          SUB   B
FFC0  C0          RET   NZ
FFC1  7D          LD    A,L
FFC2  91          SUB   C
FFC3  C9          RET

*********************************** DE:=HL-DE
                                    OUT: CY=1, wenn Übertrag
FFC4  C5          PUSH  BC               A retten (PUSH AF würde
FFC5  47          LD    B,A              Carry bei POP AF zerstören)
FFC6  7D          LD    A,L
FFC7  93          SUB   E
FFC8  5F          LD    E,A
FFC9  7C          LD    A,H
FFCA  9A          SBC   D
FFCB  57          LD    D,A
FFCC  78          LD    A,B          A wieder zurück
FFCD  C1          POP   BC
FFCE  C9          RET

*********************************** HL:=HL-DE
                                    OUT: CY=1, wenn Übertrag
FFCF  C5          PUSH  BC
FFD0  47          LD    B,A
FFD1  7D          LD    A,L
FFD2  93          SUB   E
FFD3  6F          LD    L,A
FFD4  7C          LD    A,H
FFD5  9A          SBC   D
FFD6  67          LD    H,A
```

```
FFD7  78          LD     A,B
FFD8  C1          POP    BC
FFD9  C9          RET

***********************************  BC:=HL-DE
                                     OUT: CY=1, wenn Übertrag
FFDA  E5          PUSH   HL
FFDB  67          LD     H,A
FFDC  E3          EX     (SP),HL
FFDD  7D          LD     A,L
FFDE  93          SUB    E
FFDF  4F          LD     C,A
FFE0  7C          LD     A,H
FFE1  9A          SBC    D
FFE2  47          LD     B,A
FFE3  E3          EX     (SP),HL
FFE4  7C          LD     A,H
FFE5  E1          POP    HL
FFE6  C9          RET

***********************************  HL:=HL-BC
                                     OUT: CY=1, wenn Übertrag
FFE7  D5          PUSH   DE
FFE8  57          LD     D,A
FFE9  7D          LD     A,L
FFEA  91          SUB    C
FFEB  6F          LD     L,A
FFEC  7C          LD     A,H
FFED  98          SBC    B
FFEE  67          LD     H,A
FFEF  7A          LD     A,D
FFF0  D1          POP    DE
FFF1  C9          RET

***********************************  Block nach unten verschieben
                                     (bei 664/6128 nur, wenn BC<>0)
                                     IN : HL: Zeiger auf Quellblock
                                          DE: Zeiger auf Zielblock
                                          BC: Länge
                                     OUT: HL: Zeiger nach Quellblock
                                          DE: Zeiger nach Zielblock
                                          BC: immer 0
                                          V=0, N=0, H=0
FFF2  ED B0       LDIR
FFF4  C9          RET

***********************************  Block nach oben verschieben
                                     (bei 664/6128 nur, wenn BC<>0)
                                     IN : HL: Zeiger a. Quellblockende
                                          DE: Zeiger auf Zielblockende
                                          BC: Länge
                                     OUT: HL: Zeiger vor Quellblock
                                          DE: Zeiger vor Zielblock
                                          BC: immer 0
                                          V=0, N=0, H=0
FFF5  ED B8       LDDR
FFF7  C9          RET
```

```
***********************************      JP(HL)
FFF8  E9             JP     (HL)

***********************************      JP(BC)
FFF9  C5             PUSH   BC
FFFA  C9             RET

***********************************      JP(DE)
FFFB  D5             PUSH   DE
FFFC  C9             RET

FFFD  C7             RST    00
FFFE  C7             RST    00
FFFF  54
```

6.2 Die Listings des CPC-664-ROMs

Da das ROM des 664 zu dem des 464 über weite Strecken große Ähnlichkeiten aufweist, haben wir im Folgenden nur signifikantere Änderungen der Struktur oder der Parametrisierung aufgelistet. Ein komplettes Listing des 664-ROMs wäre nur von geringem Wert und hätte den Umfang dieses Buches unverhältnismäßig erhöht (tatsächlich wären beide ROMs kaum in einem Band unterzubringen gewesen). Wenn eine Routine nicht gelistet ist, so heißt das jedoch nicht, daß sie an der gleichen Stelle liegt und exakt so aussieht, wie die entsprechende Routine im 464. Zwischen den ROMs der beiden Computer treten an fast allen Stellen durch kleinere Änderungen oder ganz neue Routinen Verschiebungen auf. Auch sind die meisten Routinen hinsichtlich der Ausnutzung des Befehlssatzes des Z80 optimiert, d.h. Sequenzen wie z.B. DEC B; JR NZ,xxxx wurden durch den Befehl DJNZ xxxx ersetzt.

Durch derartige Veränderungen, die das Programm nicht nur hinsichtlich der Ausführungszeit, sondern auch bezüglich des benötigten Speicherplatzes verbesserten, wurde es möglich, noch einige größere Routinen (neben einigen kleineren) neu aufzunehmen. Das wohl hervorstechendste Beispiel dafür ist GRA FILL. Man kann auch beobachten, daß die Veränderungen des ROMs unter anderem nach zwei Prinzipien verlief:

1. der Modularisierung, d.h. der Auslagerung einer Befehlsfolge, die des öfteren in einem Teil des Systems gebraucht wird (z.B. die Adressenberechnungs-Routine im Sound Manager),

2. dem umgekehrten Prozeß, der Demodularisierung, in der Routinen eliminiert wurden, wenn sie nur von einer Stelle aus aufgerufen werden und auch keinen allgemeineren Wert besitzen.

Man spart dadurch den Speicherplatz für das CALL, RET und gegebenenfalls noch einige PUSHes.

Die Unterschiede zwischen den beiden ROMs legen einige Vermutungen über die Personen nahe, die sie programmiert haben. So kann man - wie bereits erwähnt - viele Stellen im ROM des 464 finden, die zeigen, daß die Programmierer mit den speziellen Features des Z80 noch nicht so recht vertraut waren. Da sie sich sonst gut mit der Z80-Maschinensprache auskannten, kann man vermuten, daß sie ursprünglich 8080-Programmierer waren, die auf den Z80 umstiegen. Als dann das 664-ROM geschrieben wurde, scheinen sie sich bereits in den Z80 eingelebt gehabt zu haben, da die alten Ungeschicklichkeiten eigentlich alle eliminiert wurden.

Zum Listing des 664-ROMs sei noch gesagt, daß, wann immer wir nur einen Teil einer Routine gelistet haben, wir den Routinenkopf trotzdem vorangestellt haben. Den Namen der Routine haben wir dabei jedoch in Klammern gesetzt.

6.2.1 Das CPC 664 - Betriebssystem

Das 664-Betriebssystem ist ganz genau so strukturiert wie das des 464, wenn man davon absieht, daß das Integer Pack aus Platzgründen in den ROM-Bereich von $C000 bis $FFFF verlegt wurde. Die übrigen Packs liegen im 664 bei folgenden Adressen:

1. Kernel (KL) $0000
2. Machine Pack (MC) $057B
3. Jump Restore $08BB
4. Screen Pack (SCR) $0ABB
5. Text Screen Pack (TXT) $1070
6. Graphics Screen Pack (GRA) $15A4
7. Keyboard Manager (KM) $1B5C
8. Sound Manager (SOUND) $1FE9
9. Cassette Manager (CAS) $24BC
10. Editor (EDIT) $2C02
11. Floating Point Arithmetics (FLO) $2F7D (Zeichensatz $3800)

```
***********************************    KL EVENT
                                       IN : HL: Zeiger auf KAPQ
01E2  23         INC   HL                 Zeiger
01E3  23         INC   HL                 auf PQ-Zähler
01E4  F3         DI
01E5  7E         LD    A,(HL)           PQ-Zähler laden
01E6  34         INC   (HL)             und erhöhen
01E7  FA 01 02   JP    M,0201           $7F..$FE? d. Count dekrementieren
01EA  B7         OR    A                <>0?
01EB  20 15      JR    NZ,0202          d. erhöht lassen, raus
01ED  23         INC   HL               Zeiger auf Priorität
01EE  7E         LD    A,(HL)           Priorität laden
01EF  2B         DEC   HL               Zeiger auf PQ-Zähler
01F0  B7         OR    A                synchronous Event?
01F1  F2 2E 02   JP    P,022E           d. in SPQ einhängen
01F4  08         EX    AF,AF'           nicht im Interrupt?
01F5  30 11      JR    NC,0208          dann async. Event ausführen
01F7  08         EX    AF,AF'
01F8  87         ADD   A                Express async. Event?
01F9  F2 E8 00   JP    P,00E8           sonst in APQ einhängen
01FC  35         DEC   (HL)             PQ-Zähler wiederherstellen
01FD  23         INC   HL                 Zeiger auf
01FE  23         INC   HL                 Routinenadresse
01FF  18 21      JR    0222             und Express Event ausführen
0201  35         DEC   (HL)             PQ-Zähler wiederherstellen
0202  08         EX    AF,AF'             Interrupt
0203  38 01      JR    C,0206             nur wieder zulassen
0205  FB         EI                       wenn nicht im Interrupt
0206  08         EX    AF,AF'
0207  C9         RET

                                       (KL FIND COMMAND)
02E1  CC F1 02   CALL  Z,02F1           b1=b0=0? d. Str. in ROM suchen
02E4  DC 06 06   CALL  C,0606           gefunden? d. System starten
02E7  F1         POP   AF               Kennung aus ROM
02E8  87         ADD   A                b7=1?
02E9  30 EE      JR    NC,02D9          sonst nächstes ROM bearbeiten
02EB  79         LD    A,C              ROM-Nummer
02EC  FE 10      CP    10               innerhalb $00..$0F?
02EE  38 E9      JR    C,02D9           dann weiter durchgehen
02F0  C9         RET

***********************************    KL ROM WALK
                                       IN : DE: LoRAM
                                            HL: HiRAM
                                       OUT: DE: LoRAM, neu
                                            HL: HiRAM, neu
0326  0E 0F      LD    C,0F             Anfangs-ROM-Nummer
0328  CD 30 03   CALL  0330             ROM ggf. in VL und in Tabelle
032B  0D         DEC   C                ROM-Nummer erniedrigen
032C  F2 28 03   JP    P,0328           für ROMs 0..15 durchgehen
032F  C9         RET
```

```
***********************************   KL INIT BACK
                                      IN : DE: LoRAM
                                           HL: HiRAM
                                      OUT: DE: LoRAM, neu
                                           HL: HiRAM, neu

0330  3A D8 B8    LD     A,(B8D8)        lfd. ROM
0333  B9          CP     C               = übergebenes ROM?
0334  C8          RET    Z               dann raus
0335  79          LD     A,C             ROM-Nummer
0336  FE 10       CP     10              >$0F?
0338  D0          RET    NC              dann raus
0339  CD 79 BA    CALL   BA79            ROM anschalten
033C  3A 00 C0    LD     A,(C000)        ROM-Kennung
033F  E6 03       AND    03              b1,b0 isolieren
0341  3D          DEC    A               b1=0, b0=1?
0342  20 22       JR     NZ,0366         sonst ROM aus, raus
0344  C5          PUSH   BC              ROM-Nummer retten
0345  37          SCF                    CY:=1 als Kennzeichen
0346  CD 06 C0    CALL   C006            ROM anspringen
0349  30 1A       JR     NC,0365         Routine hat CY gelöscht? raus

***********************************   KL SCAN NEEDED
B92A 03C1  21 BF B8    LD     HL,B8BF        Ticker-Frequenzteiler
B92D 03C4  36 01       LD     (HL),01        :=1, nächster Interrupt ist Tick.
B92F 03C6  C9          RET

***********************************   Einschalt-Meldung ausgeben
0657  21 02 02    LD     HL,0202           2. Spalte, 2. Zeile
065A  CD 6C 11    CALL   116C              Cursor dorthin setzen

***********************************   (MC RESET PRINTER)
07D0  21 E7 07    LD     HL,07E7           Adresse der Default-Tabelle
07D3  11 04 B8    LD     DE,B804           Adresse der RAM-Tabelle
07D6  01 15 00    LD     BC,0015           Länge
07D9  ED B0       LDIR                     Tabelle kopieren
***********************************   Printer translation table
07E7  0A                                   Zahl der Substitutionen
07E8  A0 5E
07EA  A1 5C
07EC  A2 7B
07EE  A3 23
07F0  A6 40
07F2  AB 7C
07F4  AC 7D
07F6  AD 7E
07F8  AE 5D
07FA  AF 5B

***********************************   MC PRINT TRANSLATION
                                      IN : HL: Adresse der Tabelle
07FC  E7          RST    20                Zahl d. Substitutionen
07FD  87          ADD    A                 mal 2, da 2 Bytes pro Substitutionen
07FE  3C          INC    A                 +1 für Länge
07FF  4F          LD     C,A               Tabellenlänge nach C
0800  06 00       LD     B,00              Länge hi =0
0802  11 04 B8    LD     DE,B804           Adresse der RAM-Tabelle
0805  FE 2A       CP     2A                Tabelle nicht zu lang ?
```

```
0807  DC A1 BA    CALL  C,BAA1        dann KL LDIR, Tab. kopieren
080A  C9          RET

***********************************  MC PRINT CHAR
                                     IN : A: Zeichen
                                     OUT: CY=1, wenn o.k.
                                          CY=0, wenn busy
080B  C5          PUSH  BC
080C  E5          PUSH  HL
080D  21 04 B8    LD    HL,B804        Zeiger auf translation table
0810  46          LD    B,(HL)         Zahl der Substitutionen
0811  04          INC   B              Ausgleich für Predecrement
0812  05          DEC   B
0813  28 0A       JR    Z,081F         keine weiteren Substitutionen
0815  23          INC   HL
0816  BE          CP    (HL)           Zeichen in Tabelle enthalten ?
0817  23          INC   HL
0818  20 F8       JR    NZ,0812        nein ? dann nächstes Tabellen-Zeichen
081A  7E          LD    A,(HL)         zu substituierendes Zeichen
081B  FE FF       CP    FF             Code für Zeichen unterdrücken ?
081D  28 03       JR    Z,0822         dann nicht ausgeben
081F  CD F1 BD    CALL  BDF1           sonst MC WAIT PRINTER
0822  E1          POP   HL
0823  C1          POP   BC
0824  C9          RET

***********************************  SCR SET POSITION
                                     IN/OUT : A: SCR BASE
                                             HL: SCR OFFSET
0B41  E6 C0       AND   C0             SCR BASE, sign. Bits isolieren
0B43  32 C6 B7    LD    (B7C6),A       SCR BASE speichern
0B46  F5          PUSH  AF             und retten
0B47  7C          LD    A,H              RA-Bits
0B48  E6 07       AND   07               von SCR OFFSET
0B4A  67          LD    H,A              löschen
0B4B  CB 85       RES   0,L            Bit 0 (CCLK) löschen
0B4D  22 C4 B7    LD    (B7C4),HL      SCR OFFSET setzen
0B50  F1          POP   AF             SCR BASE hi zurück
0B51  C9          RET

***********************************  (SCR DOT POSITION)
0BC1  29          ADD   HL,HL
0BC2  19          ADD   HL,DE            Y-Koordinate mal 10

0BC3  29          ADD   HL,HL
0BC4  D1          POP   DE             X-Koordinate
0BC5  C5          PUSH  BC             RA-Bits
0BC6  CD F2 0B    CALL  0BF2           Masken holen
0BC9  78          LD    A,B            Maske für Pixelstellung im Byte
0BCA  A3          AND   E              entspricht Bits aus X-Koor. isol.
0BCB  28 05       JR    Z,0BD2         Pixel 0 im Byte ?
0BCD  CB 09       RRC   C              Auswahlmaske f. nächstes Pixel
0BCF  3D          DEC   A              Pixelnummer herunterzählen
0BD0  20 FB       JR    NZ,0BCD
0BD2  E3          EX    (SP),HL        RA-Bits vom Stack
0BD3  61          LD    H,C            Pixelauswahlmaske retten
0BD4  4D          LD    C,L            RA-Bits nach C
0BD5  E3          EX    (SP),HL        Pixelauswahlmaske auf Stack
```

```
0BD6  78          LD     A,B
0BD7  0F          RRCA

*********************************  Masken für Pixel holen
                                   OUT: B: Maske f. signifikante
                                           Bits für Pixelstellung
                                           im Byte
                                           (Bits der X-Koordinate)
0BF2  CD 08 0B    CALL   0B08         Mode-Nr. holen
0BF5  01 AA 01    LD     BC,01AA
0BF8  D8          RET    C            Mode 0 ?
0BF9  01 88 03    LD     BC,0388
0BFC  C8          RET    Z            Mode 1 ?
0BFD  01 80 07    LD     BC,0780
0C00  C9          RET

*********************************  SCR HORIZONTAL
                                   IN : DE: X-Startkoordinate
                                        BC: X-Endkoordinate
                                        HL: Y-Koordinate
                                        A: Farbmaske
0F8F  CD A9 0F    CALL   0FA9         Pen-/Linienmaske retten/setzen
0F92  CD BE 0F    CALL   0FBE         horizontale Linie ziehen
0F95  18 06       JR     0F9D         alte Pen-/Linienmaske zurück

*********************************  SCR VERTICAL
                                   IN : HL: Y-Startkoordinate
                                        BC: Y-Endkoordinate
                                        DE: X-Koordinate
                                        A: Farbmaske
0F97  CD A9 0F    CALL   0FA9         Pen-/Linienmaske retten/setzen
0F9A  CD 12 10    CALL   1012         vertikale Linie ziehen
0F9D  2A 02 B8    LD     HL,(B802)    gerettete Pen- und Linienmaske
0FA0  7D          LD     A,L          Pen-Maske
0FA1  32 A3 B6    LD     (B6A3),A     wieder setzen
0FA4  7C          LD     A,H          Linienmaske
0FA5  32 B3 B6    LD     (B6B3),A     wieder setzen
0FA8  C9          RET

*********************************  Pen- u. Linienmaske retten/setzen
                                   IN : A: neue Pen-Maske
0FA9  E5          PUSH   HL
0FAA  2A A3 B6    LD     HL,(B6A3)    Pen-Maske nach L
0FAD  32 A3 B6    LD     (B6A3),A     neue Pen-Maske setzen
0FB0  3A B3 B6    LD     A,(B6B3)     Linienmaske
0FB3  67          LD     H,A          nach H
0FB4  3E FF       LD     A,FF         Maske für durchgehende Linie
0FB6  32 B3 B6    LD     (B6B3),A     setzen
0FB9  22 02 B8    LD     (B802),HL    alte Masken zwischenspeichern
0FBC  E1          POP    HL
0FBD  C9          RET

*********************************  horizontale Linie ziehen
                                   IN : DE: X-Startkoordinate
                                        BC: X-Endkoordinate
                                        HL: Y-Koordinate
0FBE  37          SCF                 Flag für HORIZONTAL-Params
0FBF  CD 37 10    CALL   1037         Parameter holen
```

```
0FC2  CB 00       RLC   B          Linienmaske
0FC4  79          LD    A,C        Pixelauswahlmaske
0FC5  30 13       JR    NC,0FDA    Pixel nicht in Pen-Farbe ?
0FC7  1D          DEC   E          Pixelzahl lo
0FC8  20 03       JR    NZ,0FCD    weitere Pixels ?
0FCA  15          DEC   D          Pixelzahl hi
0FCB  28 2C       JR    Z,0FF9     keine weiteren Pixels ?
0FCD  CB 09       RRC   C          Auswahlmaske für nächstes Pixel
0FCF  38 28       JR    C,0FF9     Bytegrenze erreicht ?
0FD1  CB 78       BIT   7,B        nächstes Pixel
0FD3  28 24       JR    Z,0FF9     nicht in Pen-Farbe ?
0FD5  B1          OR    C          nächstes Pixel zusätzlich auswählen
0FD6  CB 00       RLC   B          Linienmaske für nächstes Pixel
0FD8  18 ED       JR    0FC7       nächstes Pixel bearbeiten
0FDA  1D          DEC   E          Pixelzahl lo
0FDB  20 03       JR    NZ,0FE0    weitere Pixels ?
0FDD  15          DEC   D          Pixelzahl hi
0FDE  28 0D       JR    Z,0FED     nicht in Pen-Farbe ?
0FE0  CB 09       RRC   C
0FE2  38 09       JR    C,0FED       sonst analog weitere Pixels
0FE4  CB 78       BIT   7,B          auswählen (wenn möglich), um
0FE6  20 05       JR    NZ,0FED      sie zeitsparend auf einmal
0FE8  B1          OR    C            zu setzen
0FE9  CB 00       RLC   B
0FEB  18 ED       JR    0FDA
0FED  C5          PUSH  BC         Linien-/Pixelauswahlmaske
0FEE  4F          LD    C,A        aktuelle Pixelauswahlmaske
0FEF  3A A4 B6    LD    A,(B6A4)   Paper-Maske
0FF2  47          LD    B,A        als Farbmaske nach B
0FF3  3A B4 B6    LD    A,(B6B4)   Hintergrund-Modus
0FF6  B7          OR    A          Z=0, wenn transparent
0FF7  18 07       JR    1000       Pixel(s) setzen
0FF9  C5          PUSH  BC         Linien-/Pixelauswahlmaske
0FFA  4F          LD    C,A        akt. Pixelauswahlmaske
0FFB  3A A3 B6    LD    A,(B6A3)   Pen-Maske
0FFE  47          LD    B,A        als Farbmaske nach B
0FFF  AF          XOR   A          Z=1 für Pixels auf jeden Fall
1000  CC E8 BD    CALL  Z,BDE8     ggf. SCR WRITE, Pixel(s) setz.
1003  C1          POP   BC         Linien-/Pixelauswahlmaske
1004  CB 79       BIT   7,C        Bytegrenze überschritten ?
1006  C4 01 0C    CALL  NZ,0C01    dann SCR NEXT BYTE
1009  7A          LD    A,D          restliche
100A  B3          OR    E            Pixelzahl
100B  20 B5       JR    NZ,0FC2    weitere Pixels ?
100D  78          LD    A,B        Linienmaske
100E  32 B3 B6    LD    (B6B3),A   für weitere Linien setzen
1011  C9          RET

**********************************  vertikale Linie ziehen
                                  IN : HL: Y-Startkoordinate
                                       BC: Y-Endkoordinate
                                       DE: X-Koordinate
1012  B7          OR    A          Flag für HORIZONTAL-Params
1013  CD 37 10    CALL  1037       Parameter holen
1016  CB 00       RLC   B          Linienmaske
1018  3A A3 B6    LD    A,(B6A3)   Pen-Maske
101B  38 09       JR    C,1026     Pixel in Pen-Farbe ?
101D  3A B4 B6    LD    A,(B6B4)   Flag für Hintergrund-Modus
```

```
1020  B7          OR    A
1021  20 09       JR    NZ,102C      Transparent-Modus ?
1023  3A A4 B6    LD    A,(B6A4)     sonst Paper-Maske
1026  C5          PUSH  BC           Linien-/Pixelauswahlmaske
1027  47          LD    B,A          Farbmaske
1028  CD E8 BD    CALL  BDE8         SCR WRITE, Pixel setzen
102B  C1          POP   BC           Linien-/Pixelauswahlmaske
102C  CD 35 0C    CALL  0C35         SCR PREV LINE
102F  1D          DEC   E
1030  20 E4       JR    NZ,1016        weitere
1032  15          DEC   D              Pixels ?
1033  20 E1       JR    NZ,1016
1035  18 D6       JR    100D         Linienmaske wieder setzen

**********************************  Linien-Parameter berechnen
                                    IN : CY=0 für vertikale Linie
                                         DE: X-Koordinate
                                         HL: Y-Koordinate
                                         BC: X- bzw. Y-Endkoordinate
                                    OUT: HL: Bildschirmadresse
                                         DE: modifizierte Länge
                                         A,B: Linienmaske
                                         C: Maske für Pixelauswahl
1037  E5          PUSH  HL           Y-Koordinate retten
1038  30 02       JR    NC,103C      vertikale Linie
103A  62          LD    H,D            sonst X-Startkoordinate
103B  6B          LD    L,E            als Startkoordinate nach HL
103C  B7          OR    A
103D  ED 42       SBC   HL,BC        Startkoordinate-Endkoordinate
103F  CD 35 19    CALL  1935         Endkoordinate-Startkoordinate
1042  24          INC   H              Länge ausgleichen wegen
1043  2C          INC   L              Predecrement
1044  E3          EX    (SP),HL      Länge retten, Y-Koord. zurück
1045  CD AB 0B    CALL  0BAB         SCR DOT POSITION, Adr. berech.
1048  3A B3 B6    LD    A,(B6B3)     Linienmaske
104B  47          LD    B,A          nach B
104C  D1          POP   DE           modifizierte Länge
104D  C9          RET

**********************************  TXT INVERSE
12C2  CD D0 BD    CALL  BDD0         TXT UNDRAW CURSOR
12C5  2A 2F B7    LD    HL,(B72F)    Paper-/Pen-Maske
12C8  7C          LD    A,H
12C9  65          LD    H,L            vertauschen
12CA  6F          LD    L,A
12CB  22 2F B7    LD    (B72F),HL    und wieder speichern
12CE  18 E3       JR    12B3         Cursor wieder zeichnen

**********************************  (TXT OUT ACTION)
1432  7E          LD    A,(HL)       Zahl der benötigten Zeichen
1433  E6 0F       AND   0F           Zahl isolieren
1435  B8          CP    B            noch nicht genug Zeichen ?
1436  D0          RET   NC           dann zurück
1437  3A 2E B7    LD    A,(B72E)     VDU-Flag
143A  A6          AND   (HL)         Flag, auch bei disabled ausg.
143B  07          RLCA               disabled und Flag nicht ges. ?
143C  38 0B       JR    C,1449       dann fertig
```

```
**********************************   TXT ASK STATE
                                     OUT: A: Status
                                            b0: Cursor enabled/disabled
                                            b1: Cursor on/off
                                            b7: VDU enabled/disabled
145C  3A 2E B7     LD     A,(B72E)
145F  C9           RET

**********************************   CHR$(0)
150F  CD A0 11     CALL   11A0         Cursor invert., Pos. prüfen
1512  C3 CD BD     JP     BDCD         Cursor wieder zurück

**********************************   (GRA RESET)
15D3  CD EC 15     CALL   15EC         diverse Initialisierungen

**********************************   GRA DEFAULT
15E8  AF           XOR    A            Force-Mode
15E9  CD 51 0C     CALL   0C51         für SCR WRITE setzen
15EC  AF           XOR    A              Hintergrund-Modus
15ED  CD D1 19     CALL   19D1           auf nicht-transparent
15F0  2F           CPL                 $FF, durchgehende Linie
15F1  CD AC 17     CALL   17AC         Flag für 1. Pixel zeichnen
15F4  C3 A8 17     JP     17A8         Linienmaske setzen

**********************************   GRA FROM USER
                                     IN : DE: X-Koordinate
                                          HL: Y-Koordinate
                                     OUT: DE: reale X-Koordinate
                                          HL: reale Y-Koordinate
1626  E5           PUSH   HL           Y-Koordinate retten
1627  CD 08 0B     CALL   0B08         Mode-Nummer holen
162A  ED 44        NEG                   A=3,1,0 für Mode 0,1,2
162C  DE FD        SBC    FD           (Maske f. nicht signif. Bits)
162E  26 00        LD     H,00

**********************************   Test, ob X-Koordinate im Window
                                     IN : DE: reale X-Koordinate
                                     OUT: CY=1, wenn im Window
                                          CY=0,Z=1,A=0, wenn zu klein
                                          CY=0,Z=0,A=$FF, wenn zu groß
1666  2A 9B B6     LD     HL,(B69B)    linke Windowgrenze
1669  37           SCF                   minus 1
166A  ED 52        SBC    HL,DE          minus X-Koordinate
166C  F2 7A 16     JP     P,167A       X-Koord. <= linke Grenze -1 ?
166F  2A 9D B6     LD     HL,(B69D)    rechte Windowgrenze
1672  B7           OR     A
1673  ED 52        SBC    HL,DE        minus X-Koordinate
1675  37           SCF                 CY=1 für im Window
1676  F0           RET    P            X-Koord. <= rechte Grenze ?
1677  F6 FF        OR     FF           Koord. zu groß,A=$FF,Z=0,CY=0
1679  C9           RET
167A  AF           XOR    A            Koord. zu klein, A=0,Z=1,CY=0
167B  C9           RET
```

```
***********************************     Test, ob Y-Koordinate im Window
                                         IN : DE: reale Y-Koordinate
                                         OUT: CY=1, wenn im Window
                                              CY=0,Z=1,A=0, wenn zu klein
                                              CY=0,Z=0,A=$FF, wenn zu groß
167C  2A 9F B6     LD      HL,(B69F)       obere Windowgrenze
167F  B7           OR      A
1680  ED 52        SBC     HL,DE           minus Y-Koordinate
1682  FA 77 16     JP      M,1677          Y-Koordinate > obere Grenze ?
1685  2A A1 B6     LD      HL,(B6A1)       untere Windowgrenze
1688  37           SCF                       minus 1
1689  ED 52        SBC     HL,DE             minus Y-Koordinate
168B  F2 7A 16     JP      P,167A          Y-Koord. <= untere Grenze-1 ?
168E  37           SCF                     sonst CY=1 für im Window
168F  C9           RET

***********************************     Test, ob Koordinate im Window, Cursor s.
                                         IN : DE: X-Koordinate
                                              HL: Y-Koordinate
                                         OUT: CY=1, wenn im Window
                                              DE: reale X-Koordinate
                                              HL: reale Y-Koordinate
                                              CY=0, wenn nicht im Window
1690  CD 23 16     CALL    1623            Cursor setzen, reale Koordinate h.
1693  E5           PUSH    HL              Y-Koordinate retten
1694  CD 66 16     CALL    1666            Test, ob X-Koordinate in Grenzen
1697  E1           POP     HL              Y-Koordinate zurück
1698  D0           RET     NC              X-Koordinate nicht in Grenzen ?
1699  D5           PUSH    DE              X-Koordinate  retten
169A  EB           EX      DE,HL
169B  CD 7C 16     CALL    167C            Test, ob Y-Koordinate in Grenzen
169E  EB           EX      DE,HL
169F  D1           POP     DE              X-Koordinate zurück
16A0  C9           RET

***********************************     GRA SET LINE MASK
                                         IN : A: Linienmaske
17A8  32 B3 B6     LD      (B6B3),A
17AB  C9           RET
***********************************     GRA SET FIRST
                                         IN : A: Flag f. 1. Pixel d. Linie
                                               $00=1. Pixel nicht zeichnen
                                               $FF=1. Pixel zeichnen
17AC  32 B2 B6     LD      (B6B2),A
17AF  C9           RET

***********************************     GRA LINE
                                         IN : DE: X-Endkoordinate
                                              HL: Y-Endkoordinate
17B0  E5           PUSH    HL
17B1  CD 87 18     CALL    1887            reale Cursorkoordinate als Startkoordinate
17B4  E1           POP     HL
17B5  CD 23 16     CALL    1623            reale Endkoordinate berechnen
17B8  E5           PUSH    HL              Y-End-Koordinate
17B9  2A A5 B6     LD      HL,(B6A5)       X-Start-Koordinate
17BC  B7           OR      A
17BD  ED 52        SBC     HL,DE           minus X-End-Koordinate
17BF  7C           LD      A,H             Vorzeichen der X-Differenz
```

```
17C0  32 AD B6     LD    (B6AD),A      speichern
17C3  FC 35 19     CALL  M,1935        Diff. negativ ? dann Betrag
17C6  D1           POP   DE            Y-End-Koordinate
17C7  E5           PUSH  HL            X-Differenz retten
17C8  2A A7 B6     LD    HL,(B6A7)     Y-Start-Koordinate
17CB  B7           OR    A
17CC  ED 52        SBC   HL,DE         minus Y-End-Koordinate
17CE  7C           LD    A,H           Vorzeichen der Y-Differenz
17CF  32 AE B6     LD    (B6AE),A      speichern
17D2  FC 35 19     CALL  M,1935        Differrenz negativ ? dann Betrag
17D5  D1           POP   DE            X-Differenz retten
17D6  B7           OR    A
17D7  ED 52        SBC   HL,DE         Y-Differenz minus X-Differenz
17D9  19           ADD   HL,DE         Y-Differenz wiederherstellen
17DA  9F           SBC   A             A=$FF, wenn X-Differenz größer
17DB  32 AF B6     LD    (B6AF),A      Flag speichern
17DE  3A AE B6     LD    A,(B6AE)      Vorzeichen der Y-Differenz
17E1  28 04        JR    Z,17E7        Y-Differenz >= X-Differenz ?
17E3  EB           EX    DE,HL         sonst Differenzen vertauschen
17E4  3A AD B6     LD    A,(B6AD)      und Vorzeichen der X-Differenz
17E7  F5           PUSH  AF            Vorzeichen der größeren Diff.
17E8  ED 53 AB B6  LD    (B6AB),DE     kleinere Differenz speichern
17EC  44           LD    B,H             größere Differenz
17ED  4D           LD    C,L             nach BC
17EE  3A B2 B6     LD    A,(B6B2)      Flag für 1. Pixel der Linie
17F1  B7           OR    A
17F2  28 01        JR    Z,17F5        1. Pixel nicht zeichnen ?
17F4  03           INC   BC            größere Differrenz erhöhen, gibt Breite
17F5  ED 43 B0 B6  LD    (B6B0),BC     größere Breite speichern
17F9  CD 35 19     CALL  1935          negative größere Differenz
17FC  E5           PUSH  HL            retten
17FD  19           ADD   HL,DE         kleinere - größere Differenz
17FE  22 A9 B6     LD    (B6A9),HL     Differenz der Diff. speichern
1801  E1           POP   HL            negative größere Differenz
1802  CB 2C        SRA   H               durch 2 f. symmetrische
1804  CB 1D        RR    L               Linie, als Vergleichswert
1806  F1           POP   AF            Vorzeichen der größeren Diff.
1807  07           RLCA
1808  38 12        JR    C,181C        negativ ?
180A  E5           PUSH  HL            Start-Vergleichswert
180B  CD 87 18     CALL  1887          neue Cursor-Koordinaten (Endk.) als Start
180E  2A AD B6     LD    HL,(B6AD)
1811  7C           LD    A,H             Vorzeichen der Differenzen
1812  2F           CPL
1813  67           LD    H,A             jeweils invertieren,
1814  7D           LD    A,L
1815  2F           CPL                   da bei Endkoordinaten
1816  6F           LD    L,A
1817  22 AD B6     LD    (B6AD),HL       angefangen wird
181A  18 12        JR    182E          keine Korrektur f. 1. Pixel
181C  3A B2 B6     LD    A,(B6B2)      Flag für 1. Pixel
181F  B7           OR    A             1. Pixel zeichnen ?
1820  20 0D        JR    NZ,182F       dann keine Korrektur
1822  19           ADD   HL,DE         kleinere Diff. zu Vergl. add.
1823  E5           PUSH  HL            neuer Vergleichswert
1824  3A AF B6     LD    A,(B6AF)      Flag für größere Differenz
1827  07           RLCA                X-Differenz größer ?
1828  DC D6 18     CALL  C,18D6        dann X-Koordinate erhöhen
```

```
182B  D4 24 19     CALL   NC,1924       sonst Y-Koordinate erhöhen
182E  E1           POP    HL            Vergleichswert
182F  7A           LD     A,D             kleinere Differenz
1830  B3           OR     E               gleich 0 ?
1831  CA 94 18     JP     Z,1894        dann nur eine Teillinie
1834  DD E5        PUSH   IX
1836  01 00 00     LD     BC,0000       größere Schrittweite =0
1839  C5           PUSH   BC              größere Schrittweite mal
183A  DD E1        POP    IX              kleinere Differenz =0
183C  DD E5        PUSH   IX              größere Schrittweite mal
183E  D1           POP    DE              kleinere Differenz nach DE
183F  B7           OR     A               als Ausgleich für alte
1840  ED 5A        ADC    HL,DE           Schrittw. addieren
1842  ED 5B AB B6  LD     DE,(B6AB)     kleinere Differenz
1846  F2 4F 18     JP     P,184F        Vergleichswert schon zu groß ?
1849  03           INC    BC            sonst größere Schritt. erhöhen
184A  DD 19        ADD    IX,DE         kleinere Diff. zu Produkt add.
184C  19           ADD    HL,DE         und zu Vergleichswert addieren
184D  30 FA        JR     NC,1849       Vergleichswert noch negativ ?
184F  AF           XOR    A               Zweierkomplement
1850  93           SUB    E               der Differenz in DE
1851  5F           LD     E,A             bilden, um Schrittweiten-
1852  9F           SBC    A               korrektur in andere Richtung
1853  92           SUB    D               zu ermöglichen
1854  57           LD     D,A
1855  19           ADD    HL,DE
1856  30 05        JR     NC,185D         Schrittweite und Schrittw.-
1858  DD 19        ADD    IX,DE           Produkt ggf. nach unten
185A  0B           DEC    BC              korrigieren
185B  18 F8        JR     1855            (abhängig v. Vergleichswert)
185D  ED 5B A9 B6  LD     DE,(B6A9)     kleinere - größere Differenz
1861  19           ADD    HL,DE         zu Vergleichswert addieren
1862  C5           PUSH   BC            größere Schrittweite
1863  E5           PUSH   HL            Vergleichswert
1864  2A B0 B6     LD     HL,(B6B0)     größere Breite (Pixelzahl)
1867  B7           OR     A               größere Schrittweite
1868  ED 42        SBC    HL,BC           subtrahieren
186A  30 06        JR     NC,1872       noch genug Pixels ?
186C  09           ADD    HL,BC         sonst alte restl. Pixelzahl
186D  44           LD     B,H             als Schrittweite bzw. Länge
186E  4D           LD     C,L             der aktuellen Teillinie
186F  21 00 00     LD     HL,0000       restliche Pixelzahl =0
1872  22 B0 B6     LD     (B6B0),HL     restliche Pixelzahl setzen
1875  CD 94 18     CALL   1894          Teillinie ziehen
1878  E1           POP    HL            Vergleichswert
1879  C1           POP    BC            größere Schrittweite
187A  30 08        JR     NC,1884       weitere Linie außerh. Window ?
187C  ED 5B B0 B6  LD     DE,(B6B0)     sonst restliche Pixelzahl
1880  7A           LD     A,D             weitere
1881  B3           OR     E               Pixels ?
1882  20 B8        JR     NZ,183C       dann nächste Teillinie ziehen
1884  DD E1        POP    IX
1886  C9           RET

**********************************    reale Cursorkoordinaten als Startkoordinaten
1887  D5           PUSH   DE
1888  CD 20 16     CALL   1620          reale Cursorkoordinaten holen
188B  ED 53 A5 B6  LD     (B6A5),DE     X-Start-Koordinate
```

```
188F  22 A7 B6     LD     (B6A7),HL     und Y-Start-Koordinate setzen
1892  D1           POP    DE
1893  C9           RET

***********************************    Teilline für GRA LINE ziehen
                                       IN : BC: Länge der Linie
                                       OUT: CY=0, wenn restliche Gesamt-
                                                  linie außerh. Window
1894  3A AF B6     LD     A,(B6AF)        Flag für größere Differenz
1897  07           RLCA                   X-Differenz größer ?
1898  38 4D        JR     C,18E7          dann horizontale Teillinie

***********************************    vertikale Teillinie ziehen
                                       IN : BC: Länge der Linie
                                       OUT: CY=0, wenn restliche Gesamt-
                                                  linie außerh. Window
189A  78           LD     A,B               Länge der Linie
189B  B1           OR     C                 =0 ?
189C  28 38        JR     Z,18D6          dann nur X-Koord. weiterzählen
189E  2A A7 B6     LD     HL,(B6A7)       laufende Y-Koordinate
18A1  09           ADD    HL,BC           Länge addieren
18A2  2B           DEC    HL              -1 gibt Endkoord. der Linie
18A3  44           LD     B,H               Y-Endkoordinate
18A4  4D           LD     C,L               nach BC
18A5  EB           EX     DE,HL           und nach DE
18A6  CD 7C 16     CALL   167C            Test, ob im Window
18A9  2A A7 B6     LD     HL,(B6A7)       Y-Start-Koordinate f. Linie
18AC  EB           EX     DE,HL           nach DE, Endkoord. nach HL
18AD  23           INC    HL              End-Koordinate +1
18AE  22 A7 B6     LD     (B6A7),HL       als nächste Start-Koordinate
18B1  38 06        JR     C,18B9          End-Koordinate im Window ?
18B3  28 21        JR     Z,18D6          unterh. Win. ? d. keine Linie
18B5  ED 4B 9F B6  LD     BC,(B69F)       sonst obere Grenze als Endwert
18B9  CD 7C 16     CALL   167C            Test, ob Startwert im Window
18BC  38 05        JR     C,18C3          Start-Koordinate im Window ?
18BE  C0           RET    NZ              oberh. Window ? d. Gesamt-Ende
18BF  ED 5B A1 B6  LD     DE,(B6A1)       untere Windowgrenze als Start
18C3  D5           PUSH   DE              Start-Koordinate retten
18C4  ED 5B A5 B6  LD     DE,(B6A5)       laufende X-Koordinate
18C8  CD 66 16     CALL   1666            liegt X-Koord. im Window ?
18CB  E1           POP    HL              Y-Start-Koordinate
18CC  38 05        JR     C,18D3          X-Koordinate im Window ?
18CE  21 AD B6     LD     HL,B6AD         Vorzeichen der X-Differenz
18D1  AE           XOR    (HL)              wird sich nicht in Window
18D2  F0           RET    P                 hineinbewegt ? d. Gesamtende
18D3  DC 12 10     CALL   C,1012          ggf. vertikale Linie ziehen
18D6  2A A5 B6     LD     HL,(B6A5)       X-Koordinate
18D9  3A AD B6     LD     A,(B6AD)        Vorzeichen der X-Differenz
18DC  07           RLCA
18DD  23           INC    HL              X-Koordinate erhöhen
18DE  38 02        JR     C,18E2          Vorzeichen negativ ?
18E0  2B           DEC    HL                sonst X-Koordinate
18E1  2B           DEC    HL                erniedrigen
18E2  22 A5 B6     LD     (B6A5),HL       X-Koordinate wieder speichern
18E5  37           SCF                    CY=1 f. kein Gesamtlinien-Ende
18E6  C9           RET
```

```
***********************************   horizontale Teillinie ziehen
                                      IN : BC: Länge der Linie
                                      OUT: CY=0, wenn Gesamtlinien-Ende
18E7  78           LD     A,B
18E8  B1           OR     C
18E9  28 39        JR     Z,1924
18EB  2A A5 B6     LD     HL,(B6A5)
18EE  09           ADD    HL,BC
18EF  2B           DEC    HL                Linie analog zu Routine
18F0  44           LD     B,H               für vertikale Teillinie
18F1  4D           LD     C,L               ($189A bis $18E6) ziehen
18F2  EB           EX     DE,HL
18F3  CD 66 16     CALL   1666
18F6  2A A5 B6     LD     HL,(B6A5)
18F9  EB           EX     DE,HL
18FA  23           INC    HL
18FB  22 A5 B6     LD     (B6A5),HL
18FE  38 06        JR     C,1906
1900  28 22        JR     Z,1924
1902  ED 4B 9D B6  LD     BC,(B69D)
1906  CD 66 16     CALL   1666
1909  38 05        JR     C,1910
190B  C0           RET    NZ
190C  ED 5B 9B B6  LD     DE,(B69B)
1910  D5           PUSH   DE
1911  ED 5B A7 B6  LD     DE,(B6A7)
1915  CD 7C 16     CALL   167C
1918  E1           POP    HL
1919  38 05        JR     C,1920
191B  21 AE B6     LD     HL,B6AE
191E  AE           XOR    (HL)
191F  F0           RET    P
1920  EB           EX     DE,HL
1921  DC BE 0F     CALL   C,0FBE
1924  2A A7 B6     LD     HL,(B6A7)
1927  3A AE B6     LD     A,(B6AE)
192A  07           RLCA
192B  23           INC    HL
192C  38 02        JR     C,1930            und Y-Koordinate analog
192E  2B           DEC    HL                weiterzählen
192F  2B           DEC    HL
1930  22 A7 B6     LD     (B6A7),HL
1933  37           SCF
1934  C9           RET

***********************************   Zweierkomplement von HL bilden
1935  AF           XOR    A
1936  95           SUB    L
1937  6F           LD     L,A
1938  9F           SBC    A
1939  94           SUB    H
193A  67           LD     H,A
193B  C9           RET

***********************************   Pixel für GRA WR CHAR setzen
                                      IN : CY: Pixel-Bit
19C0  3A A3 B6     LD     A,(B6A3)       Pen-Maske
19C3  38 08        JR     C,19CD         Pixel setzen ?
```

```
19C5  3A B4 B6     LD      A,(B6B4)      Hintergrund-Modus
19C8  B7           OR      A             transparent ?
19C9  C0           RET     NZ            dann zurück
19CA  3A A4 B6     LD      A,(B6A4)      sonst Paper-Maske
19CD  47           LD      B,A           Farbmaske nach B
19CE  C3 E8 BD     JP      BDE8          SCR WRITE, Pixel setzen

***********************************   GRA SET BACK
                                      IN : A: Hintergrund-Modus-Flag
                                              $FF = transparent
19D1  32 B4 B6     LD      (B6B4),A
19D4  C9           RET

***********************************   GRA FILL
                                      IN : A: Farbstift-Nummer
                                          HL: Bufferadresse
                                          DE: Bufferlänge
                                      OUT: CY=0, wenn Fehler
19D5  22 A5 B6     LD      (B6A5),HL     Bufferadresse speichern
19D8  36 01        LD      (HL),01       Seitenflag 1 als Bufferende
19DA  1B           DEC     DE            Bufferlänge -1 f. Endekennz.
19DB  ED 53 A7 B6  LD      (B6A7),DE     Bufferlönge speichern
19DF  CD 8A 0C     CALL    0C8A          SCR INK ENCODE, Farbmaske hol
19E2  32 AA B6     LD      (B6AA),A      Sperr-Farbmaske speichern
19E5  CD 20 16     CALL    1620          reale Cursorkoordinaten holen
19E8  CD 93 16     CALL    1693          liegen Koord. im Window ?
19EB  DC 3E 1B     CALL    C,1B3E        dann Test, ob Pixel gesetzt
19EE  D0           RET     NC            gesetzt/nicht im Window ?
19EF  E5           PUSH    HL            sonst Y-Koordinate retten
19F0  CD E3 1A     CALL    1AE3          Linie nach oben bis Sperrfarbe
19F3  E3           EX      (SP),HL       obere Y-Grenze r., alte Y-K.
19F4  CD 11 1B     CALL    1B11          Linie nach unten bis Sperrf.
19F7  C1           POP     BC            obere Y-Liniengrenze
19F8  3E FF        LD      A,FF            Flag für ausreichend Platz
19FA  32 A9 B6     LD      (B6A9),A        setzen
19FD  E5           PUSH    HL              Y-Liniengrenzen
19FE  D5           PUSH    DE              und X-Koordinate retten
19FF  C5           PUSH    BC
1A00  CD 07 1A     CALL    1A07          rechte Seite bearbeiten
1A03  C1           POP     BC
1A04  D1           POP     DE              Y-Grenzen und X-Koordinate
1A05  E1           POP     HL              zurück
1A06  AF           XOR     A             Flag für linke Seite
1A07  32 AB B6     LD      (B6AB),A      Seitenflag setzen
1A0A  CD DA 1A     CALL    1ADA          X-K. entspr. Seitenfl. veränd.
1A0D  CD 93 16     CALL    1693          Koordinaten innerh. Window ?
1A10  DC 4C 1A     CALL    C,1A4C        dann Nebenlinie bearbeiten
1A13  38 F5        JR      C,1A0A        ggf. weiter in diese Richtung
1A15  2A A5 B6     LD      HL,(B6A5)     Bufferzeiger
1A18  E7           RST     20            Seitenflag aus Buffer holen
1A19  FE 01        CP      01            Endkennzeichen ?
1A1B  28 2A        JR      Z,1A47        dann fertig
1A1D  32 AB B6     LD      (B6AB),A      Seitenflag setzen
1A20  EB           EX      DE,HL
1A21  2A A7 B6     LD      HL,(B6A7)       restliche Bufferlänge
1A24  01 07 00     LD      BC,0007         um 7 erhöhen, da 7 Bytes
1A27  09           ADD     HL,BC           pro Parameterblock im
1A28  22 A7 B6     LD      (B6A7),HL       Buffer
```

```
1A2B  EB           EX    DE,HL
1A2C  2B           DEC   HL
1A2D  E7           RST   20           obere Y-Liniengrenze
1A2E  47           LD    B,A          aus Buffer nach BC
1A2F  2B           DEC   HL
1A30  E7           RST   20
1A31  4F           LD    C,A
1A32  2B           DEC   HL
1A33  E7           RST   20           X-Koordinate aus Buffer
1A34  57           LD    D,A          nach DE
1A35  2B           DEC   HL
1A36  E7           RST   20
1A37  5F           LD    E,A
1A38  D5           PUSH  DE         X-Koordinate retten
1A39  2B           DEC   HL
1A3A  E7           RST   20           untere Y-Liniengrenze
1A3B  57           LD    D,A          aus Buffer nach DE
1A3C  2B           DEC   HL
1A3D  E7           RST   20
1A3E  5F           LD    E,A
1A3F  2B           DEC   HL
1A40  22 A5 B6     LD    (B6A5),HL  neuen Bufferzeiger setzen
1A43  EB           EX    DE,HL      untere Y-Liniengrenze nach HL
1A44  D1           POP   DE         X-Koordinate nach DE
1A45  18 C6        JR    1A0D       Linienbereich bearbeiten

1A47  3A A9 B6     LD    A,(B6A9)   Flag für Bufferüberlauf
1A4A  0F           RRCA             CY=0, wenn Bufferüberlauf
1A4B  C9           RET

**********************************   angrenzende Linien bearbeiten
                                     IN/OUT: BC: obere Y-Liniengrenze
                                             HL: untere Y-Liniengrenze
                                             DE: X-Koordinate
                                     OUT: CY=0, wenn keine Erweiterung
                                          nach dieser Seite möglich
                                     (Grenzen der Linie, die an
                                     die Linie mit den IN-Grenzen
                                     an der durch das Seitenflag
                                     bestimmten Seite angrenzt)

1A4C  ED 43 AC B6  LD    (B6AC),BC  obere Y-Liniengrenze
1A50  CD 3E 1B     CALL  1B3E       Pixel in Sperrfarben gesetzt ?
1A53  38 09        JR    C,1A5E     nein ?
1A55  CD ED 1A     CALL  1AED       nach oben bis Nicht-Sperrfarbe
1A58  D0           RET   NC         nur Sperrfarbe ?
1A59  22 AE B6     LD    (B6AE),HL  neue untere Y-Liniengrenze
1A5C  18 11        JR    1A6F       keine Erweiterung nach unten
1A5E  E5           PUSH  HL         alte untere Y-Liniengrenze
1A5F  CD 11 1B     CALL  1B11       Linie n. unten bis Sperrfarbe
1A62  22 AE B6     LD    (B6AE),HL  neue untere Y-Liniengrenze
1A65  C1           POP   BC         alte Y-Liniengrenze
1A66  7D           LD    A,L          neue Grenze < alte (unten
1A67  91           SUB   C            nach außen erweitert) ?
1A68  7C           LD    A,H          dann ggf. Params für Linie
1A69  98           SBC   B            nach anderer Seite (Linie
1A6A  DC C7 1A     CALL  C,1AC7       unter aufrufender) in Buffer
1A6D  60           LD    H,B          alte untere Y-Grenze als
```

```
1A6E   69          LD     L,C            Startposition
1A6F   CD E3 1A    CALL   1AE3          Linie nach oben bis Sperrfarbe
1A72   22 B0 B6    LD     (B6B0),HL     neue obere Y-Liniengrenze
1A75   ED 4B AC B6 LD     BC,(B6AC)     alte obere Y-Liniengrenze
1A79   B7          OR     A
1A7A   ED 42       SBC    HL,BC          alte Grenze mit neuer
1A7C   09          ADD    HL,BC          vergleichen
1A7D   28 11       JR     Z,1A90        gleich ?
1A7F   30 08       JR     NC,1A89       neue größer alte ?
1A81   CD ED 1A    CALL   1AED          nach oben bis Nicht-Sperrfarbe
1A84   DC 99 1A    CALL   C,1A99        ggf. akt. Params in Buffer
1A87   18 07       JR     1A90          neue Grenzen laden
1A89   E5          PUSH   HL             obere Y-Liniengrenze
1A8A   60          LD     H,B            nach BC
1A8B   69          LD     L,C            alte obere Y-Liniengrenze
1A8C   C1          POP    BC             nach HL, als neue untere
1A8D   CD C7 1A    CALL   1AC7          Params f. Linie unter IN-Linie
1A90   2A AE B6    LD     HL,(B6AE)     neue untere
1A93   ED 4B B0 B6 LD     BC,(B6B0)     und obere Y-Liniengrenze laden
1A97   37          SCF                  CY=1 für Bereich erweitert
1A98   C9          RET

**********************************   Parameter für Linie in Buffer
                                     IN : DE: X-Koordinate
                                          HL: untere Y-Liniengrenze
                                          BC: obere Y-Liniengrenze
                                          ($B6AB): Seitenflag
1A99   D5          PUSH   DE
1A9A   E5          PUSH   HL
1A9B   2A A7 B6    LD     HL,(B6A7)     restliche Bufferlänge
1A9E   11 F9 FF    LD     DE,FFF9        minus 7 für einen Datensatz
1AA1   19          ADD    HL,DE
1AA2   D1          POP    DE
1AA3   30 1C       JR     NC,1AC1       kein Platz mehr im Buffer ?
1AA5   22 A7 B6    LD     (B6A7),HL     neue Bufferlänge
1AA8   2A A5 B6    LD     HL,(B6A5)     Bufferzeiger
1AAB   23          INC    HL
1AAC   73          LD     (HL),E
1AAD   23          INC    HL             untere Y-Liniengrenze
1AAE   72          LD     (HL),D         in Buffer speichern
1AAF   23          INC    HL
1AB0   D1          POP    DE
1AB1   73          LD     (HL),E
1AB2   23          INC    HL             X-Koordinate
1AB3   72          LD     (HL),D         in Buffer speichern
1AB4   23          INC    HL
1AB5   71          LD     (HL),C
1AB6   23          INC    HL             obere Y-Liniengrenze
1AB7   70          LD     (HL),B         in Buffer speichern
1AB8   23          INC    HL
1AB9   3A AB B6    LD     A,(B6AB)      Seitenflag
1ABC   77          LD     (HL),A        in Buffer speichern
1ABD   22 A5 B6    LD     (B6A5),HL     neuen Bufferzeiger setzen
1AC0   C9          RET
1AC1   AF          XOR    A             Kennzeichen für Bufferüberlauf
1AC2   32 A9 B6    LD     (B6A9),A      setzen
1AC5   D1          POP    DE
1AC6   C9          RET
```

```
***********************************     Params f. andere Seite in Buffer
                                        IN : HL: neue untere Y-Grenze
                                             BC: alte untere Y-Grenze
                                             DE: X-Koordinate
1AC7  CD D3 1A     CALL   1AD3             Params für andere Seite setzen
1ACA  CD 3E 1B     CALL   1B3E             Pixel in Sperrfarbe ?
1ACD  D4 ED 1A     CALL   NC,1AED          dann n. oben bis Nicht-Sperrf.
1AD0  DC 99 1A     CALL   C,1A99           Nicht-Sperrf. ? dann in Buffer
1AD3  3A AB B6     LD     A,(B6AB)         Seitenflag
1AD6  2F           CPL                     invertieren
1AD7  32 AB B6     LD     (B6AB),A         und wieder speichern
1ADA  1B           DEC    DE               X-Koordinate erniedrigen
1ADB  3A AB B6     LD     A,(B6AB)         Seitenflag
1ADE  B7           OR     A
1ADF  C8           RET    Z                Flag für linke Seite ?
1AE0  13           INC    DE                 sonst X-Koordinate
1AE1  13           INC    DE                 erhöhen
1AE2  C9           RET

***********************************     Linie nach oben, bis Sperrfarbe
                                        IN : HL: Y-Koordinate
                                             DE: X-Koordinate
                                        OUT: HL: obere Liniengrenze
                                             DE: wie IN
                                             BC: obere Windowgrenze
1AE3  AF           XOR    A                Flag für Linie ziehen
1AE4  ED 4B 9F B6  LD     BC,(B69F)        obere Windowgrenze
1AE8  CD EF 1A     CALL   1AEF             Linie ziehen, bis Sperrfarbe
1AEB  2B           DEC    HL               Koord. unterhalb Sperrfarbe
1AEC  C9           RET

***********************************     nach oben, bis Nicht-Sperrfarbe
                                        IN : HL: Y-Koordinate
                                             BC: obere Y-Grenze
                                             DE: X-Koordinate
                                        OUT: HL: neue Y-Koordinate
                                             CY=0, wenn nur Sperrfarbe
1AED  3E FF        LD     A,FF             Flag f. Test auf Nicht-Sperrf.
1AEF  C5           PUSH   BC               obere Y-Grenze
1AF0  D5           PUSH   DE               X-Koordinate
1AF1  E5           PUSH   HL               Y-Koordinate
1AF2  F5           PUSH   AF               Flag f. ziehen/testen
1AF3  CD 4B 1B     CALL   1B4B             akt. und max. Position berech.
1AF6  F1           POP    AF               Flag f. ziehen/testen
1AF7  47           LD     B,A              nach B
1AF8  CD 30 1B     CALL   1B30             Test, ob Pixel in Sperrfarbe
1AFB  04           INC    B                  Flag für Nicht-Sperrfarbe
1AFC  10 04        DJNZ   1B02               gesucht ?
1AFE  30 47        JR     NC,1B47          Pixel in Sperrfarbe ? d. zur.
1B00  AE           XOR    (HL)               sonst Pixel in gewünschter
1B01  77           LD     (HL),A             Farbe setzen, CY=0
1B02  38 43        JR     C,1B47           Nicht-Sperrfarbe erreicht ?
1B04  E3           EX     (SP),HL            Y-Koordinate
1B05  23           INC    HL                 auf dem Stack
1B06  E3           EX     (SP),HL            erhöhen
1B07  ED 52        SBC    HL,DE            maximale Position erreicht ?
1B09  28 3C        JR     Z,1B47           dann zurück
1B0B  19           ADD    HL,DE            sonst alten Wert wiederherst.
```

```
1B0C  CD 35 0C     CALL   0C35          SCR PREV LINE, eine Zeile hoch
1B0F  18 E7        JR     1AF8          weiter prüfen

**********************************   Linie nach unten, bis Sperrfarbe
                                     IN : HL: Y-Koordinate
                                          DE: X-Koordinate
                                     OUT: HL: untere Y-Liniengrenze
                                          BC: untere Windowgrenze
1B11  C5           PUSH   BC
1B12  D5           PUSH   DE
1B13  E5           PUSH   HL
1B14  ED 4B A1 B6  LD     BC,(B6A1)     untere Windowgrenze
1B18  CD 4B 1B     CALL   1B4B          akt. und min. Pos. berechnen
1B1B  B7           OR     A
1B1C  ED 52        SBC    HL,DE         untere Grenze erreicht ?
1B1E  28 27        JR     Z,1B47        dann zurück
1B20  19           ADD    HL,DE         alten Wert wiederherstellen
1B21  CD 1B 0C     CALL   0C1B          SCR NEXT LINE, nach unten
1B24  CD 30 1B     CALL   1B30          Pixel in Sperrfarbe gesetzt ?
1B27  28 1E        JR     Z,1B47        dann fertig
1B29  AE           XOR    (HL)            sonst Pixel in gewünschter
1B2A  77           LD     (HL),A          Farbe setzen
1B2B  E3           EX     (SP),HL       X-Koordinate
1B2C  2B           DEC    HL            auf dem Stack
1B2D  E3           EX     (SP),HL       herunterzähln
1B2E  18 EB        JR     1B1B          weiter prüfen

**********************************   Test, ob Pixel in Sperrfarbe gesichert
                                     IN : HL: Bildschirmadresse
                                          C: Maske für Pixelauswahl
                                     OUT: CY=0,Z=1, wenn in Sperrfarbe
                                          A: Differenzbits zu Füllfar.
1B30  3A A3 B6     LD     A,(B6A3)      Pen-Farbmaske
1B33  AE           XOR    (HL)          mit Bildschirmbyte vergleichen
1B34  A1           AND    C             Bits für dieses Pixel isolier.
1B35  C8           RET    Z             Pixel in Pen-Farbe gesetzt ?
1B36  3A AA B6     LD     A,(B6AA)      Füll-Farbmaske
1B39  AE           XOR    (HL)          mit Bildschirmbyte vergleichen
1B3A  A1           AND    C             Bits für dieses Pixel isolier.
1B3B  C8           RET    Z             Pixel in Füll-Farbe gesetzt ?
1B3C  37           SCF                  sonst CY=1, nicht in Sperrfar.
1B3D  C9           RET

**********************************   Test, ob Pixel in Sperrfarbe gesichert
                                     IN : HL: Y-Koordinate
                                          DE: X-Koordinate
                                     OUT: CY=0,Z=1, wenn in Sperrfarbe
                                          A: Differenzbits zu Füllfar.
1B3E  C5           PUSH   BC
1B3F  D5           PUSH   DE
1B40  E5           PUSH   HL
1B41  CD AB 0B     CALL   0BAB          Bildschirmadr. und Maske ber.
1B44  CD 30 1B     CALL   1B30          testen, ob Pixel in Sperrfarbe
1B47  E1           POP    HL
1B48  D1           POP    DE
1B49  C1           POP    BC
1B4A  C9           RET
```

```
*********************************   aktuelle und Grenz-Position berechnen
                                    IN : DE: X-Koordinate
                                         HL: akt. Y-Koordinate
                                         BC: Grenz-Y-Koordinate
                                    OUT: HL: akt. Bildschirmadresse
                                         DE: Grenz-Bildschirmadresse
                                         C: Maske für Pixelauswahl
1B4B  C5          PUSH  BC             Grenz-Y-Koordinate
1B4C  D5          PUSH  DE             X-Koordinate
1B4D  CD AB 0B    CALL  0BAB           Bildschirmadr. der akt. Pos.
1B50  D1          POP   DE             X-Koordinate zurück
1B51  E3          EX    (SP),HL        Grenz-Y-Koordinate zurück
1B52  CD AB 0B    CALL  0BAB           Bildschirmadr. der Grenzpos.
1B55  EB          EX    DE,HL          nach DE
1B56  E1          POP   HL             akt. Bildschirmadr. nach HL
1B57  C9          RET

*********************************   KM FLUSH
1BFE  CD C5 1B    CALL  1BC5           Zeichen lesen
1C01  38 FB       JR    C,1BFE         bis kein Zeichen in Buffer
1C03  C9          RET

*********************************   KM SET LOCKS
                                    IN : L<b7>=1 f. Shift Lock
                                         H<b7>=1 f. Caps Lock
1D3C  22 31 B6    LD    (B631),HL      Caps/Shift Lock Flags setzen
1D3F  C9          RET

*********************************   Parameter-Blöcke initialisieren
                                    IN : C: Kanalbits
201A  DD E5       PUSH  IX
201C  E5          PUSH  HL
201D  21 F0 B1    LD    HL,B1F0          Scan Sound Queues
2020  34          INC   (HL)             ausschalten
2021  E5          PUSH  HL             Zeiger auf Flag retten
2022  DD 21 B9 B1 LD    IX,B1B9        Params Kanal A -$3F
2026  79          LD    A,C            Kanalbits
2027  CD 09 22    CALL  2209           Adresse der Params berechnen
202A  F5          PUSH  AF             restl. Kanalbits retten
202B  C5          PUSH  BC
202C  CD 86 22    CALL  2286           Kanal aus lfd. Aktiv. löschen
202F  CD E7 23    CALL  23E7           und ausschalten
2032  DD E5       PUSH  IX             Zeiger auf Params
2034  D1          POP   DE             nach DE
2035  13          INC   DE
2036  13          INC   DE               Zeiger auf Kanalstatus
2037  13          INC   DE
2038  6B          LD    L,E              nach HL
2039  62          LD    H,D              kopieren
203A  13          INC   DE             Zeiger auf ENT-Flag
203B  01 3B 00    LD    BC,003B        Länge d. restl. Param-Blocks
203E  36 00       LD    (HL),00        Status löschen
2040  ED B0       LDIR                 und restl. Block löschen
2042  DD 36 1C 04 LD    (IX+1C),04     4 freie Plätze in Queue
2046  C1          POP   BC
2047  F1          POP   AF             Kanalbits zurück
2048  20 DD       JR    NZ,2027        noch Kanäle? d. initialisieren
204A  E1          POP   HL               Bearbeitungsflag wieder auf
```

```
204B  35           DEC   (HL)            alten Wert setzen
204C  E1           POP   HL
204D  DD E1        POP   IX
204F  C9           RET

***********************************  SOUND CONTINUE
206B  11 ED B1     LD    DE,B1ED       Zeiger auf alte Aktivitäten
206E  1A           LD    A,(DE)        alte Aktivitäten laden
206F  B7           OR    A             keine alten Aktivitäten?
2070  C8           RET   Z             dann raus
2071  D5           PUSH  DE            Zeiger retten
2072  DD 21 B9 B1  LD    IX,B1B9       Params Kanal A -$3F
2076  CD 09 22     CALL  2209          Adresse d. alten Aktiv. ber.
2079  F5           PUSH  AF            restl. Aktiv. retten
207A  DD 7E 0F     LD    A,(IX+0F)     lfd. Lautstärke laden
207D  DC DE 23     CALL  C,23DE        und in entspr. PSG-Reg. (!)
2080  F1           POP   AF            restl. alte Aktivitäten
2081  20 F3        JR    NZ,2076       ggf. nächsten Kanal bearbeiten
2083  E3           EX    (SP),HL       Zeiger alte Aktiv. -> HL
2084  7E           LD    A,(HL)        alte Aktivitäten
2085  36 00        LD    (HL),00       löschen
2087  23           INC   HL            und als lfd. Aktivitäten
2088  77           LD    (HL),A        setzen
2089  E1           POP   HL
208A  C9           RET

***********************************  Sound Event (asynchronous)
208B  DD E5        PUSH  IX
208D  3A EE B1     LD    A,(B1EE)      lfd. Aktivitäten
2090  B7           OR    A             keine?
2091  28 3D        JR    Z,20D0        dann raus
2093  F5           PUSH  AF            Aktivitäten retten
2094  DD 21 B9 B1  LD    IX,B1B9       Params Kanal A -$3F
2098  01 3F 00     LD    BC,003F       Länge eines Parameter-Blockes
209B  DD 09        ADD   IX,BC         addieren
209D  CB 3F        SRL   A             bis ein 1-Bit
209F  30 FA        JR    NC,209B       gefunden
20A1  F5           PUSH  AF            restliche Kanal-Bits retten
20A2  DD 7E 04     LD    A,(IX+04)     ENT-Flag
20A5  1F           RRA                 ENT-Folge zu bearbeiten?
20A6  DC 1F 24     CALL  C,241F        dann ENT-Folge bearbeiten
20A9  DD 7E 07     LD    A,(IX+07)     ENV-Flag
20AC  1F           RRA                 ENV-Folge zu bearbeiten?
20AD  DC 1F 23     CALL  C,231F        dann ENV-Folge bearbeiten
20B0  DC 13 22     CALL  C,2213        ggf. nächsten Ton bearbeiten
20B3  F1           POP   AF            restl. lfd. Aktivitäten
20B4  20 E2        JR    NZ,2098       ggf. nächsten Kanal bearbeiten
20B6  C1           POP   BC            lfd. Aktivitäten nach B
20B7  3A EE B1     LD    A,(B1EE)      alte Aktivitäten
20BA  2F           CPL                 aus
20BB  A0           AND   B             lfd. Aktivitäten herausnehmen
20BC  28 12        JR    Z,20D0        keine Aktiv. übrig? d. raus
20BE  DD 21 B9 B1  LD    IX,B1B9       Params Kanal A -$3F
20C2  11 3F 00     LD    DE,003F       Länge eines Parameterblockes
20C5  DD 19        ADD   IX,DE         addieren
20C7  CB 3F        SRL   A             unterstes Bit ins Carry
20C9  F5           PUSH  AF            und restl. Kanal-Bits retten
20CA  DC E7 23     CALL  C,23E7        Kanal übrig? dann ausschalten
```

```
20CD  F1            POP   AF           restl. Kanal-Bits
20CE  20 F5         JR    NZ,20C5      ggf. nächsten Kanal bearbeiten
20D0  AF            XOR   A            Flag für 'Kanäle bearbeiten'
20D1  32 F0 B1      LD    (B1F0),A     löschen
20D4  DD E1         POP   IX
20D6  C9            RET

********************************** (SOUND QUEUE)
2142  B0            OR    B            und noch als Rendezvous setzen
2143  E6 0F         AND   0F           u. isolieren, 'hold' in b3
2145  4F            LD    C,A          neuer Status nach C
2146  E5            PUSH  HL           Zeiger auf Übergabe retten
2147  21 F0 B1      LD    HL,B1F0        Scan Sound Queues
214A  34            INC   (HL)           verhindern
214B  E3            EX    (SP),HL      Zeiger auf Übergabe nach HL
214C  23            INC   HL           Zeiger auf ENV-Folgenummer
214D  DD 21 B9 B1   LD    IX,B1B9      Params Kanal A -$003F
2151  11 3F 00      LD    DE,003F      Länge eines Parameterblockes

2192  C1            POP   BC           Kanalbits u. Datenstatus
2193  E1            POP   HL           Zeiger auf Übergabe
2194  04            INC   B            zum Ausgleich erhöhen
2195  10 BA         DJNZ  2151         ggf. weitere Kanäle bearbeiten
2197  E3            EX    (SP),HL      Zeiger auf Bearbeitungsflag
2198  35            DEC   (HL)         Bearbeitungsflag wiederherst.
2199  E1            POP   HL
219A  37            SCF                CY:=1 für o.k.
219B  C9            RET

********************************** (SOUND RELEASE)
21B4  21 F0 B1      LD    HL,B1F0        Scan Sound Queues
21B7  34            INC   (HL)           verhindern
21B8  E5            PUSH  HL           Zeiger auf Bearbeitungsflag
21B9  DD 21 B9 B1   LD    IX,B1B9      Params Kanal A -$3F
21BD  CD 09 22      CALL  2209         Adresse d. Parameterblocks
21C0  F5            PUSH  AF           restliche Kanal-Bits retten
21C1  DD CB 03 5E   BIT   3,(IX+03)    Queue im Haltezustand?
21C5  C4 19 22      CALL  NZ,2219      d. Kanal aktivieren
21C8  F1            POP   AF           restliche Kanal-Bits
21C9  20 F2         JR    NZ,21BD      ggf. weitere Kanäle bearbeiten
21CB  E1            POP   HL           Zeiger auf Bearbeitungsflag
21CC  35            DEC   (HL)         Flag wiederherstellen
21CD  C9            RET

********************************** Adresse eines Param-Blocks ber.
                                   IN : IX: Adr. alter Param-Block
                                         A: Kanal-Bits,<>0 (wichtig!)
                                   OUT: IX: Adr. neuer Param-Block
                                         A: neue Kanalbits
                                         CY:=1
2209  11 3F 00      LD    DE,003F      Länge eines Parameter-Blockes
220C  DD 19         ADD   IX,DE        addieren
220E  CB 3F         SRL   A            bis ein 1-Bit
2210  D8            RET   C            gefunden
2211  18 F9         JR    220C         sonst weiter addieren
```

```
**********************************   (Kanal aktivieren)
2254  F1            POP    AF           ENV-Nummer
2255  CD DE 22      CALL   22DE         Dauer, Rauschen und ENV setzen
2258  21 EE B1      LD     HL,B1EE      Zeiger lfd. Aktivitäten
225B  DD 46 01      LD     B,(IX+01)    lfd. Kanalmaske
225E  7E            LD     A,(HL)       lfd. Aktivitäten laden
225F  B0            OR     B            Kanal in lfd. Aktivitäten
2260  77            LD     (HL),A       und lfd. Aktivitäten neu setz.
2261  A8            XOR    B            sonstige Kanäle
2262  20 03         JR     NZ,2267      aktiv?
2264  23            INC    HL             sonst 100Hz Frequenzteiler
2265  36 03         LD     (HL),03        auf Startwert setzen
2267  DD 34 19      INC    (IX+19)      lfd. Datenblocknummer erhöhen

**********************************   Queue anhalten, Kanal aus Aktiv.
                                     IN : IX: Zeiger auf Params
                                          HL: Zeiger lfd. Datenblock
2280  CB 9E         RES    3,(HL)       Hold-Bit in Daten ausschalten
2282  DD 36 03 08   LD     (IX+03),08   und Kanal in Haltezustand

**********************************   Kanal aus lfd. Aktivit. löschen
                                     IN : IX: Zeiger Params
2286  21 EE B1      LD     HL,B1EE      Zeiger auf lfd. Aktivitäten
2289  DD 7E 01      LD     A,(IX+01)    Kanalmaske
228C  2F            CPL                   Kanal
228D  A6            AND    (HL)           aus lfd. Aktivitäten
228E  77            LD     (HL),A         löschen
228F  C9            RET

**********************************   Kanal abschalten
                                     IN : IX: Zeiger Params
23E7  AF            XOR    A            Maske für Kanal abschalten

**********************************   (CAS SET SPEED)
                                     OUT: A: beim Lesen erkannter
                                             Baudratenwert
24DC  3A E7 B1      LD     A,(B1E7)     Wert laden
24DF  37            SCF                 CY=1 für o.k.
24E0  C9            RET

**********************************   (CAS OUT OPEN)
                                     IN : HL: Adresse des Filenamens
                                          B: Länge des Filenamens
                                          DE: Zeiger auf Ausgabebuffer
                                     OUT: HL: Zeiger auf gen. Header
                                          CY=0 für Fehler
                                          dann A=$0E für schon offen
24FE  DD 21 5F B1   LD     IX,B15F      Zeiger auf Ausgabeparameter
2502  DD 7E 00      LD     A,(IX+00)    Filestatus
2505  B7            OR     A
2506  3E 0E         LD     A,0E         Nummer für Filestatus-Fehler
2508  C0            RET    NZ           schon offen ? dann Fehler

**********************************   CAS IN CLOSE
                                     OUT: CY=0 für Fehler
                                          dann A=$0E für nicht offen
2550  3A 1A B1      LD     A,(B11A)     Eingabefile-Status
2553  B7            OR     A
```

```
2554  3E 0E      LD    A,0E        Nummer für Filestatus-Fehler
2556  C8         RET   Z           nicht offen ? dann Fehler

***********************************  CAS IN ABANDON
                                   OUT: CY=1 (immer)
                                        A=$FF für alle Files geschl.
2557  21 1A B1   LD    HL,B11A       Zeiger auf Eingabefilestatus
255A  06 01      LD    B,01          Bit für Eingabeflag
255C  7E         LD    A,(HL)        alten Filestatus laden
255D  36 00      LD    (HL),00       Filestatus auf geschlossen
255F  C5         PUSH  BC            Bit für Ein-/Ausgabeflag
2560  CD 6D 25   CALL  256D          ggf. Buffer löschen
2563  F1         POP   AF            Bit für Ein-/Ausgabeflag
2564  21 E4 B1   LD    HL,B1E4       Zeiger auf Ein-/Ausgabeflag
2567  AE         XOR   (HL)          entsprechendes Bit invertieren
2568  37         SCF                 CY=1 für o.k.
2569  C0         RET   NZ            weitere Kennz. gesetzt ?
256A  77         LD    (HL),A        Ein-/Ausgabeflag auf inaktiv
256B  9F         SBC   A             A=$FF
256C  C9         RET

***********************************  Ein-/Ausgabebuffer ggf. löschen
                                   IN : A: alter Filestatus
                                        HL: Zeiger auf Status
256D  FE 04      CP    04               nicht zeichenweise Datei
256F  D8         RET   C                oder CATALOG ? dann zurück
2570  23         INC   HL            Zeiger auf Bufferparameter
2571  5E         LD    E,(HL)           Bufferadresse
2572  23         INC   HL               nach DE
2573  56         LD    D,(HL)
2574  6B         LD    L,E              und nach HL als Quelle
2575  62         LD    H,D
2576  13         INC   DE            Bufferadr. +1 als Ziel
2577  36 00      LD    (HL),00       1. Byte löschen
2579  01 FF 07   LD    BC,07FF       Länge
257C  C3 A1 BA   JP    BAA1          KL LDIR, Buffer löschen

***********************************  CAS OUT CLOSE
                                   OUT: CY=0 für Fehler
                                        dann A=$0E, Z=0 für n. offen
                                             A=$00, Z=1 für Abbruch
257F  3A 5F B1   LD    A,(B15F)      Ausgabefile-Status
2582  FE 03      CP    03            Abbruch ?
2584  28 13      JR    Z,2599        dann keinen Block speichern
2586  C6 FF      ADD   FF
2588  3E 0E      LD    A,0E          Nummer für Status-Fehler
258A  D0         RET   NC            nicht offen ? dann Fehler
258B  21 75 B1   LD    HL,B175          Kennzeichen für letzten
258E  35         DEC   (HL)             Block setzen
258F  23         INC   HL
2590  23         INC   HL
2591  7E         LD    A,(HL)           Block-
2592  23         INC   HL               länge
2593  B6         OR    (HL)             <>0 ?
2594  37         SCF                 CY=1 für kein Abbruch
2595  C4 86 27   CALL  NZ,2786       dann Block speichern
2598  D0         RET   NC            Abbruch ?
```

```
***********************************  CAS OUT ABANDON
                                     OUT: CY=1 (immer)
                                          A=$FF für alle Files geschl.
2599  21 5F B1    LD     HL,B15F        Zeiger auf Ausgabefilestatus
259C  06 02       LD     B,02           Bit für Ausgabeflag
259E  18 BC       JR     255C

***********************************  (Block von Kassette lesen)
                                     OUT: CY=0, Z=0, A=$0F für EOF
                                          CY=0, Z=1, für Abbruch
                                          CY=1 für o.k
26AC  3A 30 B1    LD     A,(B130)       Flag für letzen Block
26AF  B7          OR     A
26B0  3E 0F       LD     A,0F           Nummer für EOF
26B2  C0          RET    NZ             letzter Block ? dann Fehler

***********************************  (EDIT)
                                     IN : HL: Zeiger auf Eingabebuffer
                                     OUT: HL: Zeiger auf Eingabebuffer
                                          Z=1, wenn mit ESC terminiert
2C02  C5          PUSH   BC
2C03  D5          PUSH   DE
2C04  E5          PUSH   HL
2C05  CD F2 2D    CALL   2DF2           Copy Cursor auschalten
2C08  01 FF 00    LD     BC,00FF        Pos. in Buffer und Bufferlänge
2C0B  7E          LD     A,(HL)         Zeichen aus Buffer
2C0C  FE 30       CP     30             <"0"?
2C0E  38 07       JR     C,2C17         dann keine Ziffer, raus
2C10  FE 3A       CP     3A             <="9"?
2C12  DC 42 2C    CALL   C,2C42         dann Ziffer, übernehmen
2C15  38 F4       JR     C,2C0B         und ggf. nächstes Zeichen
2C17  78          LD     A,B            Position in Buffer
2C18  B7          OR     A              nicht am Bufferanfang?
2C19  7E          LD     A,(HL)         lfd. Zeichen aus Buffer
2C1A  C4 42 2C    CALL   NZ,2C42        ggf. übernehmen
2C1D  E5          PUSH   HL             lfd. Zeiger als Bufferanfang
2C1E  0C          INC    C              Bufferlänge erhöhen
2C1F  7E          LD     A,(HL)         Zeichen aus Buffer

***********************************  Zeichen in Buffer übernehmen
2C42  0C          INC    C              Bufferlänge erhöhen
2C43  04          INC    B              Position in Buffer erhöhen
2C44  23          INC    HL             Zeiger auf nächstes Zeichen
2C45  C3 25 2F    JP     2F25           Zeichen ausgeben

***********************************  Copy Cursor ausgeschaltet?
                                     OUT: Z=1, wenn CC ausgeschaltet
2EC1  2A 16 B1    LD     HL,(B116)      CC Koordinaten
2EC4  7C          LD     A,H              testen
2EC5  B5          OR     L
2EC6  C9          RET
```

```
***********************************    Mantissenvergleich
                                       IN : DE,HL: Mant1
                                            (IY): Mant2
                                            B: 1. MSB Mant2
                                       OUT: Z=1, wenn gleich
                                            CY=0, wenn Mant1 größer
369D  7A          LD    A,D
369E  B8          CP    B                   Mantissen
369F  C0          RET   NZ                  byteweise
36A0  7B          LD    A,E                 vergleichen
36A1  FD BE 02    CP    (IY+02)
36A4  C0          RET   NZ
36A5  7C          LD    A,H
36A6  FD BE 01    CP    (IY+01)
36A9  C0          RET   NZ
36AA  7D          LD    A,L
36AB  FD BE 00    CP    (IY+00)
36AE  C9          RET
```

6.2.2 Das Basic des CPC 664

```
*********************************** (AUTO-Zeile auswerten)
C095  CD 52 DE    CALL  DE52       Spaces, TABs und LFs überlesen
C098  CD D4 EE    CALL  EED4       Zeilennummern-String wandeln
C09B  30 0A       JR    NC,C0A7    Fehler (keine Zeilennummer) ?
C09D  CD 52 DE    CALL  DE52       Spaces, TABs und LFs überlesen
C0A0  B7          OR    A          Zeilenende ?
C0A1  37          SCF
C0A2  CC 69 E8    CALL  Z,E869     dann Zeile im Programm suchen
C0A5  30 E3       JR    NC,C08A    nicht gefunden ? dann fertig
C0A7  D4 DE C0    CALL  NC,C0DE    keine Zeilen-Nr. ? dann AUTO absch.
C0AA  21 8A AC    LD    HL,AC8A    Zeiger auf Zeilen-Buffer
C0AD  18 08       JR    C0B7

*********************************** normale Zeile holen/auswerten
C0AF  CD FC CA    CALL  CAFC       Eingabezeile holen
C0B2  30 FB       JR    NC,C0AF    Abbruch ? dann neue Zeile
C0B4  CD 9B C3    CALL  C39B       Linefeed ausgeben
C0B7  CD 52 DE    CALL  DE52       Spaces, TABs und LFs überlesen
C0BA  B7          OR    A          Zeilenende ?
C0BB  28 CA       JR    Z,C087     dann neue Zeile
C0BD  CD D4 EE    CALL  EED4       Zeilennummern-String wandeln
C0C0  30 0B       JR    NC,C0CD    keine Zeilennummer ?
C0C2  CD 4D FB    CALL  FB4D       Strings in String-Bereich forcieren
C0C5  CD AA E7    CALL  E7AA       Zeile im Programm einfügen
C0C8  CD 8F C1    CALL  C18F       Basic-Zeiger initialisieren
C0CB  18 BA       JR    C087       nächste Zeile holen

*********************************** Eingabezeile für AUTO holen
                                   OUT: CY=0 für Abbruch
                                        HL: Zeiger auf Zeile
C10D  2A 02 AC    LD    HL,(AC02)  aktuelle AUTO-Zeilennummer
C110  EB          EX    DE,HL      nach DE
C111  D5          PUSH  DE         und retten
C112  CD 3D E2    CALL  E23D       Zeile/Zeilennr. nach ASCII
C115  CD DE C0    CALL  C0DE       AUTO (zunächst) ausschalten
C118  CD 04 CB    CALL  CB04       Zeile ausgeben, neue Z. holen
C11B  D1          POP   DE         Zeilennummer
C11C  D0          RET   NC         Abbruch ? dann zurück
C11D  E5          PUSH  HL         Zeiger auf Zeile
C11E  2A 04 AC    LD    HL,(AC04)  AUTO-Schrittweite
C121  19          ADD   HL,DE      zu Zeilennummer addieren
C122  D4 E1 C0    CALL  NC,C0E1    ggf. AUTO-Znr./Flag setzen
C125  E1          POP   HL         Zeiger auf Zeile
C126  37          SCF              CY=1 für keinen Abbruch
C127  C9          RET

*********************************** Basic-Befehl CLEAR INPUT
C13F  CD 31 DE    CALL  DE31       INPUT-Token übergehen
C142  C3 3D BD    JP    BD3D       KM FLUSH, Tastaturbuffer leeren

*********************************** opt. Eingabekanal transp. setzen
C1D7  CD FE C1    CALL  C1FE       optionale Filenummer holen
C1DA  CD B6 C1    CALL  C1B6       als Eingabekanalnummer setzen
C1DD  C1          POP   BC         Aufrufadresse
```

```
C1DE  F5          PUSH   AF        alte Eingabekanalnummer
C1DF  CD C7 C1    CALL   C1C7      aktuelle Kanalnummer holen
C1E2  CD F0 C1    CALL   C1F0      als Streamnr., Rout. weiterf.
C1E5  F1          POP    AF        alte Eingabekanalnummer
C1E6  18 CE       JR     C1B6      wieder setzen

********************************** Bytewert <2 (als Flag) holen
                                 OUT: A: Bytewert
C223  3E 02       LD     A,02        Limit+1
C225  18 EF       JR     C216        Bytewert <2 holen

********************************** Basic-Befehl PEN
C227  CD E8 C1    CALL   C1E8        opt. Streamnr. transp. setzen
C22A  01 90 BB    LD     BC,BB90     TXT SET PEN
C22D  C4 42 C2    CALL   NZ,C242     kein Komma ? dann PEN setzen
C230  CD 46 DE    CALL   DE46        folgt Komma ?
C233  D0          RET    NC          nein ? dann zurück
C234  CD 23 C2    CALL   C223        Bytewert <2 holen
C237  01 9F BB    LD     BC,BB9F     TXT SET BACK
C23A  18 09       JR     C245        Hintergrund-Modus setzen

********************************** Basic-Befehl PAPER
C23C  CD E8 C1    CALL   C1E8        opt. Streamnr. transp. setzen
C23F  01 96 BB    LD     BC,BB96     TXT SET PAPER
C242  CD 74 C2    CALL   C274        Farbstiftnummer holen
C245  E5          PUSH   HL          Basic-PC
C246  CD FC FF    CALL   FFFC        Routine ausführen
C249  E1          POP    HL          Basic-PC
C24A  C9          RET

********************************** Window-Nr. transparent setzen
C28C  CD 10 C2    CALL   C210        Filenr. holen
C28F  FE 08       CP     08          Nummer >=8 ?
C291  30 8D       JR     NC,C220     dann "Improper argument"
C293  F5          PUSH   AF          Nummer retten
C294  CD 22 DE    CALL   DE22        Test auf Klammer zu
C297  F1          POP    AF          Window-Nummer
C298  C3 EF C1    JP     C1EF        transparent setzen

********************************** Basic-Funktion COPYCHR$
C29B  CD 8C C2    CALL   C28C        Window-Nr. transparent setzen
C29E  CD 60 BB    CALL   BB60        Zeichen an Cursorpos. lesen
C2A1  C3 78 FA    JP     FA78        in 1-Zeichen-String wandeln

********************************** Basic-Befehl CURSOR
C363  CD E8 C1    CALL   C1E8        opt. Streamnr. transp. setzen
C366  28 0A       JR     Z,C372      folgt Komma ?
C368  CD 23 C2    CALL   C223        Bytewert <2 holen
C36B  B7          OR     A           Byte =0 ?
C36C  CC 84 BB    CALL   Z,BB84      dann TXT CUR OFF
C36F  C4 81 BB    CALL   NZ,BB81     sonst TXT CUR ON
C372  CD 46 DE    CALL   DE46        folgt Komma ?
C375  D0          RET    NC          nein ?
C376  CD 23 C2    CALL   C223        sonst Byte <2 holen
C379  B7          OR     A           Byte =0 ?
C37A  CA 7E BB    JP     Z,BB7E      dann TXT CUR DISABLE
C37D  C3 7B BB    JP     BB7B        sonst TXT CUR ENABLE
```

```
C44F  C3 3A CC     JP     CC3A          "Broken in" ausg., DERR setzen

**********************************  Basic-Funktion EOF
C452  E5           PUSH   HL               Basic-PC
C453  CD 89 BC     CALL   BC89             CAS TEST EOF
C456  28 F7        JR     Z,C44F           Abbruch ? dann Fehler melden
C458  3F           CCF
C459  9F           SBC    A                A=$FF bei EOF, sonst A=0
C45A  CD 2D FF     CALL   FF2D             Byte nach FAC
C45D  E1           POP    HL               Basic-PC
C45E  C9           RET

**********************************  Zeichen von Kassette lesen
C45F  CD 80 BC     CALL   BC80             CAS IN CHAR
C462  D8           RET    C                kein Fehler ?
C463  28 EA        JR     Z,C44F           Abbruch ? dann DERR setzen
C465  EE 0E        XOR    0E               Filestatus-Fehler ?
C467  C0           RET    NZ               nein ? dann zurück
C468  CD 48 CB     CALL   CB48             "File not open" ausgeben
C46B  1F                                   Nr. für "File not open"

**********************************  ESC-Abbruch ggf. ermöglichen
C482  E5           PUSH   HL
C483  C5           PUSH   BC
C484  D5           PUSH   DE
C485  11 95 C4     LD     DE,C495          Adresse der Event-Routine
C488  0E FD        LD     C,FD             ROM-Konf., Basic-ROM ein
C48A  3A 0B AC     LD     A,(AC0B)         Flag für "ON BREAK CONT"
C48D  B7           OR     A                nicht gesetzt ?
C48E  C4 45 BB     CALL   NZ,BB45          dann KM ARM BREAK
C491  D1           POP    DE
C492  C1           POP    BC
C493  E1           POP    HL
C494  C9           RET

**********************************  nach ESC auf weitere Taste warten
                                    OUT : CY=1 für Abbruch
C4A4  C5           PUSH   BC
C4A5  D5           PUSH   DE
C4A6  E5           PUSH   HL
C4A7  CD B6 BC     CALL   BCB6             SOUND HOLD
C4AA  F5           PUSH   AF               Flag für Kanäle aktiv retten
C4AB  CD 40 BD     CALL   BD40             TXT ASK STATE
C4AE  47           LD     B,A              Cursor-/VDU-Status retten
C4AF  CD 81 BB     CALL   BB81             Cursor einschalten
C4B2  CD 06 BB     CALL   BB06             auf Taste warten
C4B5  FE EF        CP     EF               BRK-Code ?
C4B7  28 F9        JR     Z,C4B2           dann neue Taste
C4B9  CB 48        BIT    1,B              war Cursor vorher OFF ?
C4BB  C4 84 BB     CALL   NZ,BB84          dann Cursor wieder OFF
C4BE  FE FC        CP     FC               Code für ESC ?
C4C0  37           SCF                     CY=1 für Abbruch
C4C1  28 0B        JR     Z,C4CE           ESC ?
C4C3  FE 20        CP     20               Space ?
C4C5  C4 0C BB     CALL   NZ,BB0C          nein ? dann zurück in Buffer
C4C8  F1           POP    AF               Flag für Kanal-Aktivitäten
C4C9  F5           PUSH   AF               Stack-Ausgleich (entspr. ESC)
C4CA  DC B9 BC     CALL   C,BCB9           ggf. SOUND CONTINUE
```

```
C4CD  B7             OR     A            CY=0 für keinen Abbruch
C4CE  E1             POP    HL           Aktivitäts-Flag löschen
C4CF  E1             POP    HL
C4D0  D1             POP    DE
C4D1  C1             POP    BC
C4D2  C9             RET

**********************************    Basic-Befehl ON BREAK CONT
C4D3  AF             XOR    A            Flag für ESC gesperrt
C4D4  18 02          JR     C4D8         setzen

**********************************    ON BREAK CONT ausschalten
C4D6  3E FF          LD     A,FF         Flag für ESC nicht gesperrt
C4D8  32 0B AC       LD     (AC0B),A     Flag speichern
C4DB  E5             PUSH   HL
C4DC  CD 48 BB       CALL   BB48         KM DISARM BREAK, ESC verrieg.
C4DF  18 A2          JR     C483         ggf. ESC wieder ermöglichen

**********************************    Basic-Befehl FILL
C515  CD 74 C2       CALL   C274         Farbstiftnummer holen
C518  E5             PUSH   HL           Basic-PC retten
C519  F5             PUSH   AF           Farbstiftnr. retten
C51A  CD 64 FC       CALL   FC64         Garbage collection
C51D  CD 08 F7       CALL   F708         Größe/Adr. des freien Platzes
C520  01 1D 00       LD     BC,001D      Mindest-Platz
C523  CD DE FF       CALL   FFDE         mit freiem Platz vergleichen
C526  3E 07          LD     A,07         Nr. für "Memory full"
C528  DA 58 CB       JP     C,CB58       kein Platz ? dann Fehler
C52B  EB             EX     DE,HL        Ende der Felder nach DE
C52C  F1             POP    AF           Farbstiftnummer
C52D  CD 52 BD       CALL   BD52         GRA FILL, Fläche ausfüllen
C530  E1             POP    HL           Basic-PC
C531  C9             RET

**********************************    Ansprung der Graphikbefehle
                                      IN : BC: Routinenadresse
C54E  C5             PUSH   BC           Routinenadresse retten
C54F  CD 8F C5       CALL   C58F         Graphik-Koordinaten holen
C552  CD 46 DE       CALL   DE46         folgt Komma ?
C555  30 05          JR     NC,C55C      nein ?
C557  FE 2C          CP     2C           folgt zweites Komma ?
C559  C4 BD C5       CALL   NZ,C5BD      nein ? dann Graphik-Pen holen
C55C  CD 46 DE       CALL   DE46         folgt Komma ?
C55F  30 0A          JR     NC,C56B      nein ?
C561  3E 04          LD     A,04         Limit+1
C563  CD 16 C2       CALL   C216         Byte <4 holen
C566  E5             PUSH   HL           Basic-PC
C567  CD 59 BC       CALL   BC59         SCR ACCESS, Zeichenmodus setz.
C56A  E1             POP    HL           Basic-PC
C56B  E3             EX     (SP),HL      retten, Routinenadr. zurück
C56C  C5             PUSH   BC           2. Koordinate
C56D  E3             EX     (SP),HL      nach HL, Routinenadresse
C56E  C1             POP    BC           nach BC
C56F  CD FC FF       CALL   FFFC         Routine ausführen
C572  E1             POP    HL           Basic-PC wieder zurück
C573  C9             RET
```

```
**********************************  Basic-Befehl GRAPHICS
C59D  FE BA       CP    BA            folgt Token für PAPER ?
C59F  28 13       JR    Z,C5B4        dann GRAPHICS PAPER

**********************************  Basic-Befehl GRAPHICS PEN
C5A1  CD 2A DE    CALL  DE2A          Test auf PEN
C5A4  BB                              Token für PEN
C5A5  FE 2C       CP    2C            folgt kein Komma ?
C5A7  C4 BD C5    CALL  NZ,C5BD       dann GRAPHICS PEN setzen
C5AA  CD 46 DE    CALL  DE46          folgt Komma ?
C5AD  D0          RET   NC            nein ? dann zurück
C5AE  CD 23 C2    CALL  C223          sonst Wert <2 holen
C5B1  C3 46 BD    JP    BD46          Graphik-Hintergrundmodus setz.

**********************************  Basic-Befehl GRAPHICS PAPER
C5B4  CD 31 DE    CALL  DE31          PAPER-Token übergehen
C5B7  CD 74 C2    CALL  C274          Farbstiftnr. holen
C5BA  C3 E4 BB    JP    BBE4          GRA SET PAPER

C5BD  CD 74 C2    CALL  C274          Farbstiftnr. holen
C5C0  C3 DE BB    JP    BBDE          GRA SET PEN

**********************************  Basic-Befehl MASK
C5C3  FE 2C       CP    2C            folgt Komma ?
C5C5  28 06       JR    Z,C5CD        dann keine Linienmaske
C5C7  CD BB CE    CALL  CEBB          Bytewert als Linienmaske holen
C5CA  CD 4C BD    CALL  BD4C          GRA SET LINE MASK
C5CD  CD 46 DE    CALL  DE46          folgt Komma ?
C5D0  D0          RET   NC            nein ? dann zurück
C5D1  CD 23 C2    CALL  C223          Bytewert <2 holen
C5D4  C3 49 BD    JP    BD49          GRA SET FIRST, Flag 1. Pixel

**********************************  Basic-Befehl ON BREAK
C979  CD 7F C9    CALL  C97F          Befehl ausführen
C97C  C3 31 DE    JP    DE31          nächstes Zeichen holen

C97F  FE 8B       CP    8B            folgt Token für CONT ?
C981  CA D3 C4    JP    Z,C4D3        dann ON BREAK CONT
C984  FE CE       CP    CE            folgt Token für STOP ?
C986  11 00 00    LD    DE,0000       Kennz. für ON BREAK inaktiv
C989  28 08       JR    Z,C993        STOP ?
C98B  CD 2A DE    CALL  DE2A          sonst Test auf GOSUB
C98E  9F                              Token für GOSUB
C98F  CD 2C E8    CALL  E82C          Zeilenadresse holen
C992  2B          DEC   HL            PC eins zurück
C993  ED 53 1A AC LD    (AC1A),DE     Adr. der ON BREAK-Routine
C997  C3 D6 C4    JP    C4D6          ON BREAK CONT abschalten

**********************************  Eingabezeile für LINE INPUT holen
CAEF  CD FC CA    CALL  CAFC          Eingabezeile holen
CAF2  D8          RET   C             kein Abbruch ?
CAF3  CD A4 C1    CALL  C1A4          sonst I/O-Kanäle init.
CAF6  31 00 C0    LD    SP,C000       Stackpointer init.
CAF9  C3 62 DE    JP    DE62          Befehl nochmals ausführen

**********************************  Fehler entspr. Byte nach Aufruf
CB48  E3          EX    (SP),HL       Aufrufadr. nach HL
CB49  7E          LD    A,(HL)        Byte nach Aufruf als Fehlernr.
```

```
CB4A  18 0C        JR     CB58          Fehler behandeln

*********************************   Ausgabe von "Syntax error"
CB4C  3E 02        LD     A,02          Nr. für "Syntax error"
CB4E  18 08        JR     CB58          Fehler behandeln

*********************************   Ausgabe von "Improper argument"
CB50  3E 05        LD     A,05          Nr. für "Improper argument"
CB52  18 04        JR     CB58          Fehler behandeln

*********************************   (Fehlerbehandlung)
CB8E  36 00        LD     (HL),00       Flag f. ON ERROR-Rout. inaktiv
CB90  3A 90 AD     LD     A,(AD90)      Fehlernummer
CB93  CD 8F CE     CALL   CE8F          Adresse d. Fehlerstrings holen
CB96  2A 8C AD     LD     HL,(AD8C)     Adresse der Fehlerzeile
CB99  CD B2 DE     CALL   DEB2          als aktuelle Zeilennr. setzen
CB9C  3A 90 AD     LD     A,(AD90)      Fehlernummer
CB9F  EE 20        XOR    20
CBA1  20 04        JR     NZ,CBA7       nicht "Broken in" ?
CBA3  3A 91 AD     LD     A,(AD91)      Ein-/Ausgabefehlercode (DERR)
CBA6  17           RLA                  Fehler bereits mitgeteilt ?
CBA7  D4 07 CC     CALL   NC,CC07       nein ? dann Fehler ausgeben
CBAA  C3 58 C0     JP     C058          zur Eingabeschleife

*********************************   DERR setzen, "Broken in" melden
CC3A  32 91 AD     LD     (AD91),A      Ein-/Ausgabefehl. (DERR) setz.
CC3D  CD 48 CB     CALL   CB48          Fehler ausgeben
CC40  20                                Nr. für "Broken in"

*********************************   Parameter für CALL/RSX holen
                                    OUT: DE: Integerwert bzw.
                                             Descriptoradresse
CEE6  CD 65 CF     CALL   CF65          Ausdruck holen
CEE9  CD 66 FF     CALL   FF66          Typ des FAC holen
CEEC  20 0D        JR     NZ,CEFB       kein String ? d. Integer n. DE
CEEE  E5           PUSH   HL            Basic-PC
CEEF  2A A0 B0     LD     HL,(B0A0)     Adresse des Descriptors
CEF2  CD 58 FB     CALL   FB58          String in Stringbereich forc.
CEF5  EB           EX     DE,HL         Descriptoradresse nach DE
CEF6  E1           POP    HL            Basic-PC
CEF7  C9           RET

*********************************   Tabelle der Hierarchiecodes und
                                    der Operatorenadressen
CFF0  0C
CFF1  0C FD        FD0C                 +, Addition (numerisch)

CFF3  0C
CFF4  21 FD        FD21                 -, Subtraktion

CFF6  12
CFF7  35 FD        FD35                 *, Multiplikation

CFF9  12
CFFA  52 FD        FD52                 /, Division

CFFC  16
CFFD  39 D5        D539                 ^, Potenzierung
```

```
CFFF  10
D000  67 FD        FD67               \, Integerdivision

D002  06
D003  87 FD        FD87               AND

D005  0E
D006  79 FD        FD79               MOD

D008  04
D009  92 FD        FD92               OR

D00B  02
D00C  9C FD        FD9C               XOR

D00E  0A
D00F  11 D0        D011               numerischer Vergleich

*********************************** Basic-Funktion DERR
D12E  3A 91 AD     LD     A,(AD91)    Ein-/Ausgabefehler
D131  18 03        JR     D136        in FAC eintragen

*********************************** Variablenadresse nach FAC ("§")
D151  CD CC D6     CALL   D6CC        Variable holen, Adresse n. DE
D154  D2 50 CB     JP     NC,CB50     existiert Var. n. ? d. Fehler
D157  E5           PUSH   HL          Basic-PC
D158  EB           EX     DE,HL       Variablenadresse nach HL
D159  78           LD     A,B         Typ der Variablen
D15A  FE 03        CP     03          String ?
D15C  CC 58 FB     CALL   Z,FB58      dann in Stringbereich forcier.
D15F  CD 89 FE     CALL   FE89        Adresse in positive REAL-Zahl
D162  E1           POP    HL          Basic-PC
D163  C9           RET

*********************************** (Basic-Befehl CAT)
D2A1  CD 9B BC     CALL   BC9B        CAS CATALOG
D2A4  CA 3A CC     JP     Z,CC3A      Abbruch ? dann "Broken in"

*********************************** Basic-Befehl OPENOUT
D2AB  CD CA D2     CALL   D2CA        File transparent öffnen

D2AE  CD 25 F7     CALL   F725        Ausgabebuffer belegen
D2B1  CD 6C C4     CALL   C46C        Position für Kassette/Disk. =1
D2B4  C3 8C BC     JP     BC8C        CAS OUT OPEN

*********************************** (File öffnen)
D2CA  CD 2A F7     CALL   F72A        Buffer reservieren
D2CD  CD 06 CF     CALL   CF06        Filenamen holen

D2D2  CD DE D2     CALL   D2DE        File eröffnen
D2D5  CA 3A CC     JP     Z,CC3A      Abbruch ? dann "Broken in"

*********************************** (Basic-Befehl CLOSEOUT)
D2F9  CD 8F BC     CALL   BC8F        CAS OUT CLOSE
D2FC  CA 3A CC     JP     Z,CC3A      Abbruch ? dann "Broken in"
```

```
***********************************  def. Funkt. und Var.-Offs. lösch.
D611  21 00 00     LD     HL,0000       Null als
D614  22 EB AD     LD     (ADEB),HL     1. Offset der VL d. Funktionen
D617  C3 52 EA     JP     EA52          Variablenoffsets löschen

***********************************  Word vom Basic-Stack holen
                                     OUT: HL: Word
D92B  3E 02        LD     A,02          2 Bytes
D92D  CD 65 F6     CALL   F665          vom Basic-Stack
D930  7E           LD     A,(HL)
D931  23           INC    HL              Word aus Basic-Stack
D932  66           LD     H,(HL)          nach HL
D933  6F           LD     L,A
D934  C9           RET

***********************************  Zeile für LINE INPUT holen
                                     OUT: HL: Zeiger auf Zeile
DB36  CD C7 C1     CALL   C1C7          akt. Eingabekanalnr. holen
DB39  D2 5C DC     JP     NC,DC5C       Kassette/Diskette ?
DB3C  CD EF CA     CALL   CAEF          Zeile holen, bis kein Abbruch
DB3F  3A 14 AE     LD     A,(AE14)      Linefeed-Flag
DB42  FE 3B        CP     3B
DB44  C4 9B C3     CALL   NZ,C39B       ggf. Linefeed ausgeben
DB47  C9           RET

***********************************  Text für INPUT holen und ausgeben
DBB6  CD 79 F8     CALL   F879          String holen, auf Stringstack
DBB9  CD C7 C1     CALL   C1C7          akt. Eingabekanalnr. holen
DBBC  D2 F5 FB     JP     NC,FBF5       Kass./Disk. ? dann vom Stack
DBBF  C3 D0 F8     JP     F8D0          sonst String ausgeben

DD2F bis DE19: INTEGER ARITHMETICS
(liegt im CPC 464 von 3708 bis 37FE)

***********************************  Test auf Komma
DE1A  3E 2C        LD     A,2C          ","
DE1C  18 10        JR     DE2E

***********************************  Test auf Klammer auf
DE1E  3E 28        LD     A,28          "("
DE20  18 0C        JR     DE2E

***********************************  Test auf Klammer zu
DE22  3E 29        LD     A,29          ")"
DE24  18 08        JR     DE2E

***********************************  Test auf "="
DE26  3E EF        LD     A,EF          Token für "="
DE28  18 04        JR     DE2E

***********************************  Zeile/Nr. für AUTO nach ASCII
                                     IN : DE:  Zeilennummer
E23D  CD 69 E8     CALL   E869          Zeile im Programm suchen
E240  38 17        JR     C,E259        gefunden ? dann nach ASCII
E242  EB           EX     DE,HL         Zeilennummer nach HL
E243  CD 4F EF     CALL   EF4F          Zeilennummer nach ASCII
E246  11 00 01     LD     DE,0100       Zähler f. restl. Bufferlänge
E249  01 8A AC     LD     BC,AC8A       Zeiger auf Buffer
```

```
E24C  7E            LD     A,(HL)       Zeichen aus ASCII-Zeilennummer
E24D  23            INC    HL
E24E  02            LD     (BC),A       in Buffer kopieren
E24F  03            INC    BC
E250  15            DEC    D            restliche Bufferlänge
E251  B7            OR     A
E252  20 F8         JR     NZ,E24C      noch kein Zeilennr.-Ende ?
E254  02            LD     (BC),A       sonst Null ans neue Znr.-Ende
E255  0B            DEC    BC           Zeiger davor
E256  C3 ED E2      JP     E2ED         Space nach Zeilennr. in Buffer

**********************************    REM-Token nach ASCII wandeln
E2F1  CD 01 E3      CALL   E301         REM-Token nach ASCII
E2F4  7E            LD     A,(HL)       Zeichen aus Zeile
E2F5  B7            OR     A
E2F6  C8            RET    Z            Zeilenende ?
E2F7  CD CF E2      CALL   E2CF         Zeichen so übernehmen
E2FA  23            INC    HL
E2FB  18 F7         JR     E2F4         nächstes Zeichen

**********************************    Keyword-Token nach ASCII wandeln
E2FD  FE C5         CP     C5           Token für REM ?
E2FF  28 F0         JR     Z,E2F1       dann inkl. restl. Zeile übern.

**********************************    Bereich aus Programm löschen
                                      IN : HL: Adresse des Bereichs
                                           BC: Länge des Bereichs
E7E9  78            LD     A,B
E7EA  B1            OR     C            Länge =0 ?
E7EB  C8            RET    Z            dann fertig
E7EC  EB            EX     DE,HL        Löschadresse nach DE
E7ED  CD F1 F6      CALL   F6F1         Bereich löschen
E7F0  C3 0C F6      JP     F60C         Programm-/Var.-Zeiger korrig.

**********************************    Basic-Befehl REM
E9AC  7E            LD     A,(HL)
E9AD  B7            OR     A
E9AE  C8            RET    Z              nächstes Zeilenende suchen
E9AF  23            INC    HL
E9B0  18 FA         JR     E9AC

**********************************    REM bzw. "'" überlesen
EA4A  F5            PUSH   AF           Token retten
EA4B  23            INC    HL           Zeiger nach Token
EA4C  CD AC E9      CALL   E9AC         Zeilenende suchen
EA4F  F1            POP    AF           Token
EA50  2B            DEC    HL           Zeiger vor Zeilenende
EA51  C9            RET

**********************************    (Basic-Befehl RUN)
EAB0  E1            POP    HL
EAB1  E3            EX     (SP),HL
EAB2  CD 43 BD      CALL   BD43         GRA DEFAULT
EAB5  E1            POP    HL
EAB6  23            INC    HL
EAB7  C3 7C DE      JP     DE7C
```

```
**********************************   Programmzeichen einlesen
                                     OUT: A: Zeichen
                                             CY=0 für Fehler
EC01  CD 80 BC     CALL   BC80          CAS IN CHAR
EC04  D8           RET    C             kein Fehler ?
EC05  FE 1A        CP     1A            erstmaliges EOF ?
EC07  37           SCF                  dann kein Fehler,
EC08  C8           RET    Z             zurück
EC09  32 91 AD     LD     (AD91),A      sonst Nummer für DERR retten
EC0C  3F           CCF                  CY=0 für Fehler
EC0D  C9           RET

**********************************   EOF melden
                                     IN : A: Fehlernr. für DERR
EC0E  32 91 AD     LD     (AD91),A      Fehlernr. für DERR speichern
EC11  CD 6F C1     CALL   C16F          Basic initialisieren
EC14  3E 18        LD     A,18          Nr. für "EOF met"
EC16  F5           PUSH   AF            Fehlernr. retten
EC17  CD 03 D3     CALL   D303          Kassette/Disk abbrechen
EC1A  F1           POP    AF            Fehlernr.
EC1B  C3 58 CB     JP     CB58          Fehler melden

**********************************   Dezimalpunkt u. Exponenten setzen
                                     IN : C: Gesamtkommastellenzahl
                                          D: Formatierungsflags
                                          E: Kommaposition
                                     OUT: B: Zahl der Vorkommastellen
EF9B  7A           LD     A,D           Formatierungsflags
EF9C  87           ADD    A
EF9D  30 2D        JR     NC,EFCC       keine spezielle Formatierung ?
EF9F  FA F2 EF     JP     M,EFF2        form. Exponentialdarstellung ?
EFA2  7B           LD     A,E           Kommaposition
EFA3  81           ADD    C             + Stellenzahl = dez. Exp. +1
EFA4  D6 15        SUB    15            Dezimalexponent < 20 ?
EFA6  FA 5B F0     JP     M,F05B        dann o.k.
EFA9  7A           LD     A,D             sonst Flag für Format-
EFAA  F6 41        OR     41              überlauf und Exponential-
EFAC  57           LD     D,A             darstellung setzen
EFAD  18 43        JR     EFF2          format. Exponentialdarstellung

**********************************   (formatierte Exponentialdarst.)
F00E  CB 4A        BIT    1,D
F010  28 07        JR     Z,F019        keine Komma-Einteilung ?
F012  78           LD     A,B           Vorkommastellenzahl
F013  04           INC    B             Ausgleich für Predecrement
F014  05           DEC    B             Stelle für "," abziehen
F015  D6 04        SUB    04            3 Stellen und "," abziehen
F017  30 FB        JR     NC,F014       weitere Kommata ?

**********************************   ggf. führende Null in Buffer
                                     IN : D: Formatierungsflags
                                          E: Vorzeichen
                                     IN/OUT: HL: Bufferzeiger
F131  E5           PUSH   HL
F132  7E           LD     A,(HL)        Zeichen aus Zahl
F133  23           INC    HL
F134  3D           DEC    A
F135  FE 30        CP     30            "0" ?
```

```
F137  38 F9      JR    C,F132      dann nächstes Zeichen prüfen
F139  3C         INC   A           Zeichen wiederherstellen
F13A  20 01      JR    NZ,F13D     kein Ende der Zahl ?
F13C  5F         LD    E,A         sonst Vorzeichen auf positiv
F13D  E1         POP   HL          Bufferzeiger auf Zahl
F13E  7A         LD    A,D         Formatierungsflags
F13F  EE 80      XOR   80          formatierte Darstellung ?
F141  F4 00 F1   CALL  P,F100      dann Vork.-Stellenz. holen
F144  D8         RET   C             zu viele Sonder-
F145  C8         RET   Z             zeichen ?
F146  3E 30      LD    A,30        sonst "0"
F148  18 06      JR    F150        führend in Buffer

**********************************  ggf. führendes Währungszeichen s.
                                    IN/OUT: B: Vorkommastellenzahl
                                            D: Formatierungsflags
                                            HL: Bufferzeiger
F14A  CB 52      BIT   2,D         Flag für Währungszeichen
F14C  C8         RET   Z           nicht gesetzt ?
F14D  3A 54 AE   LD    A,(AE54)    sonst Währungszeichen
F150  04         INC   B           Vorkomma-Stellenzahl erhöhen
F151  2B         DEC   HL          Zeiger vor Zahl
F152  77         LD    (HL),A      Zeichen vor die Zahl setzen
F153  C9         RET

**********************************  Vorzeichen setzen
                                    IN/OUT: E: Vorzeichen
                                            D: Formatierungsflags
                                            HL: Bufferzeiger
F154  7B         LD    A,E         Vozeichen
F155  87         ADD   A           ins Carry
F156  3E 2D      LD    A,2D        "-"
F158  38 0E      JR    C,F168      negativ ?
F15A  7A         LD    A,D           Vorzeichen vor der Zahl,
F15B  E6 98      AND   98            kein "+" vor der Zahl und
F15D  EE 80      XOR   80            formatierte Darstellung ?
F15F  C8         RET   Z           dann kein Vorzeichen
F160  E6 08      AND   08          Flag f. "+" bei pos. Vorzeich.
F162  3E 2B      LD    A,2B        "+"
F164  20 02      JR    NZ,F168     Flag gesetzt ?
F166  3E 20      LD    A,20        sonst Space
F168  CB 62      BIT   4,D         Flag für Vorzeichen nach Zahl
F16A  28 E4      JR    Z,F150      nein ? dann vor die Zahl
F16C  32 50 AE   LD    (AE50),A    Vorzeichen nach Zahl setzen
F16F  AF         XOR   A           Null
F170  32 51 AE   LD    (AE51),A    als Endkennzeichen
F173  C9         RET

**********************************  gepackte BCD-Zahl nach ASCII
                                    IN : B: Zahl der BCD-Bytes
                                        DE: Zeiger auf BCD-Zahl
                                    OUT: C: Länge der ASCII-Zahl
                                        HL: Zeiger auf ASCII-Zahl
F1C6  21 50 AE   LD    HL,AE50     Zeiger auf ASCII-Bufferende
F1C9  36 00      LD    (HL),00     Null ans Bufferende
F1CB  78         LD    A,B         Länge der BCD-Zahl in Bytes
F1CC  87         ADD   A           mal 2
F1CD  4F         LD    C,A         gibt Zahl der ASCII-Ziffern
```

```
F1CE  C8          RET   Z          keine BCD-Ziffern ?
F1CF  3E 30       LD    A,30       Hi-Nibble der Zifferncodes =3
F1D1  EB          EX    DE,HL      ASCII-Zg. n. DE, BCD-Zg. n. HL
F1D2  ED 67       RRD              näch. BCD-Stelle ins Lo-Nibble
F1D4  1B          DEC   DE         ASCII-Bufferzeiger
F1D5  12          LD    (DE),A     ASCII-Ziffer abspeichern
F1D6  ED 67       RRD              näch. BCD-Stelle ins Lo-Nibble
F1D8  1B          DEC   DE         ASCII-Bufferzeiger
F1D9  12          LD    (DE),A     ASCII-Ziffer abspeichern
F1DA  23          INC   HL         Zeiger auf nächstes BCD-Byte
F1DB  10 F5       DJNZ  F1D2       weitere BCD-Bytes ?
F1DD  EB          EX    DE,HL      ASCII-Bufferzeiger nach HL
F1DE  FE 30       CP    30         führende Ziffer ="0" ?
F1E0  C0          RET   NZ         nein ?
F1E1  0D          DEC   C            sonst Null
F1E2  23          INC   HL           unterdrücken
F1E3  C9          RET

**********************************  Zahl nach Hex-/Binär-String wand.
                                  IN : HL: Zahl
                                       A: Mindest-Stellenzahl
                                       B: Zahl der Bits pro Stelle
                                       C: Bitmaske für eine Stelle
                                       (BC=$0101 für Binärzahl,
                                       BC=$040F für Hex-Zahl)
                                  OUT: HL: Zeiger auf String
F1E4  D5          PUSH  DE
F1E5  EB          EX    DE,HL        Zahl nach DE
F1E6  21 3E AE    LD    HL,AE3E      Zeiger auf Bufferende
F1E9  36 00       LD    (HL),00      Null ans Bufferende
F1EB  3D          DEC   A            Stellenzähler erniedrigen

**********************************  (Parameter holen, Routine ausf.)
F279  C5          PUSH  BC
F27A  CD E6 CE    CALL  CEE6         Integer bzw. Descr.-Adr. holen
F27D  C1          POP   BC

**********************************  PRINT, Ausdruck ausgeben
F2D7  CD 65 CF    CALL  CF65         Ausdruck holen
F2DA  F5          PUSH  AF           Flag für Statementende
F2DB  E5          PUSH  HL           und Basic-PC retten
F2DC  CD 66 FF    CALL  FF66           Typ des Ausdrucks
F2DF  28 0F       JR    Z,F2F0         String ?
F2E1  CD 6D EF    CALL  EF6D         FAC nach ASCII wandeln
F2E4  CD 8A F8    CALL  F88A         String auf Stringstack
F2E7  36 20       LD    (HL),20      String mit Space abschließen
F2E9  2A A0 B0    LD    HL,(B0A0)    Zeiger auf Descriptor
F2EC  34          INC   (HL)         Länge für Space erhöhen
F2ED  7E          LD    A,(HL)       Stringlänge
F2EE  18 1F       JR    F30F         String ausgeben

F2F0  2A A0 B0    LD    HL,(B0A0)    Zeiger auf Descriptor
F2F3  46          LD    B,(HL)       Stringlänge
F2F4  0E 00       LD    C,00         Zähler f. druckbare Zeichen =0
F2F6  23          INC   HL
F2F7  7E          LD    A,(HL)         Stringadresse
F2F8  23          INC   HL             nach HL
F2F9  66          LD    H,(HL)
```

```
F2FA  6F        LD     L,A
F2FB  04        INC    B             Ausgleich für Predecrement
F2FC  18 0E     JR     F30C
F2FE  7E        LD     A,(HL)        Zeichen aus String
F2FF  FE 20     CP     20
F301  23        INC    HL
F302  30 07     JR     NC,F30B       kein Steuerzeichen ?
F304  3D        DEC    A               nicht CHR$(01) (Steuerzei-
F305  20 07     JR     NZ,F30E         chen für direkte Ausgabe) ?
F307  05        DEC    B             Länge für folgendes Zeichen
F308  28 04     JR     Z,F30E        String zu Ende ?
F30A  23        INC    HL
F30B  0C        INC    C             Zahl der druckb. Zeichen erh.
F30C  10 F0     DJNZ   F2FE          weitere Zeichen im String ?
F30E  79        LD     A,C           Zahl der druckbaren Zeichen
F30F  CD EA C2  CALL   C2EA          paßt String noch in Zeile ?
F312  D4 9B C3  CALL   NC,C39B       nein ? dann Linefeed ausgeben
F315  CD D0 F8  CALL   F8D0          String ausgeben, vom Stack
F318  E1        POP    HL            Basic-PC
F319  F1        POP    AF            Flag für Statementende ?
F31A  CC 9B C3  CALL   Z,C39B        dann Linefeed ausgeben
F31D  C9        RET

*********************************** (Formatstring auswerten)
F491  13        INC    DE              nächstes
F492  05        DEC    B               Zeichen
F493  28 0E     JR     Z,F4A3        keine weiteren Zeichen ?
F495  1A        LD     A,(DE)        Zeichen laden
F496  CD 07 F5  CALL   F507          Währungszeichen ?
F499  20 08     JR     NZ,F4A3       nein ?
F49B  24        INC    H             sonst Vorkommastellenzahl erh.
F49C  2E 24     LD     L,24          Flag f. "**" und Währungsz.
F49E  32 54 AE  LD     (AE54),A      Währungszeichen speichern
F4A1  13        INC    DE              nächstes
F4A2  05        DEC    B               Zeichen
F4A3  79        LD     A,C
F4A4  B5        OR     L               Formatierungsflags
F4A5  4F        LD     C,A             entsprechend setzen

*********************************** auf Währungszeichen prüfen
                                    IN : A: Zeichen
                                    OUT: Z=1, wenn Währungszeichen
F507  FE 24     CP     24            Dollar-Zeichen ("$") ?
F509  C8        RET    Z
F50A  FE A3     CP     A3            Pfund-Zeichen ?
F50C  C9        RET

*********************************** Basic-Befehl MEMORY
F570  CD F8 CE  CALL   CEF8          Adresse holen als neues HIMEM
F573  E5        PUSH   HL            Basic-PC retten
F574  2A 60 AE  LD     HL,(AE60)     Ende des freien RAMs
F577  CD D8 FF  CALL   FFD8          kleiner als neues HIMEM ?
F57A  38 31     JR     C,F5AD        dann "Memory full"
F57C  13        INC    DE            neues HIMEM+1
F57D  CD F1 F5  CALL   F5F1          größer als alter Wert ?
F580  DC 8F F5  CALL   C,F58F        dann auf Platz prüfen
F583  EB        EX     DE,HL         neues HIMEM+1 nach HL
F584  CD 08 F8  CALL   F808          HIMEM neu setzen
```

```
F587  2A 76 B0     LD     HL,(B076)     Adresse der Ein-/Ausgabebuffer
F58A  22 78 B0     LD     (B078),HL     für Freigabe speichern
F58D  E1           POP    HL            Basic-PC
F58E  C9           RET

**********************************   auf Platz oberhalb HIMEM prüfen
                                      IN : DE: neuer Wert für HIMEM+1
F58F  CD AE BB     CALL   BBAE          Params d. User-Matrizen holen
F592  ED 4B 5E AE  LD     BC,(AE5E)     HIMEM-Zeiger
F596  DC E5 F5     CALL   C,F5E5        Wert innerh. HIMEM...User-M. ?
F599  38 12        JR     C,F5AD        nein ? dann "Memory full"
F59B  2A 76 B0     LD     HL,(B076)     Adresse der Ein-/Ausgabebuffer
F59E  2B           DEC    HL
F59F  CD E5 F5     CALL   F5E5          Wert innerh. HIMEM...Buffer ?
F5A2  D0           RET    NC            dann o.k.
F5A3  3A 75 B0     LD     A,(B075)      Ein-/Ausgabebuffer-Status
F5A6  B7           OR     A             keine Buffer reserviert ?
F5A7  C8           RET    Z             dann o.k.
F5A8  FE 04        CP     04            reserviert, aber unbenutzt ?
F5AA  CA 7F F7     JP     Z,F77F        dann freigeben, o.k.
F5AD  C3 75 F8     JP     F875          sonst "Memory full"

**********************************   Test auf Platz für Binärdatei
                                      IN : DE: Startadresse
                                           BC: Länge
F5B0  D5           PUSH   DE            Startadresse retten
F5B1  EB           EX     DE,HL         und nach HL
F5B2  09           ADD    HL,BC         Länge addieren
F5B3  2B           DEC    HL            -1 gibt Endadresse
F5B4  ED 4B 62 AE  LD     BC,(AE62)     Start des freien RAMs (LoRAM)
F5B8  E3           EX     (SP),HL       Endadr. retten, Start zurück
F5B9  EB           EX     DE,HL         Startadresse nach DE
F5BA  2A 5E AE     LD     HL,(AE5E)     HIMEM-Zeiger
F5BD  CD E5 F5     CALL   F5E5          Start innerh. LoRAM...HIMEM ?
F5C0  EB           EX     DE,HL
F5C1  E3           EX     (SP),HL         Startadresse retten,
F5C2  EB           EX     DE,HL           Endadresse zurück
F5C3  DC E5 F5     CALL   C,F5E5        Endad. innerh. LoRAM...HIMEM ?
F5C6  30 E5        JR     NC,F5AD       dann "Memory full"
F5C8  ED 4B 76 B0  LD     BC,(B076)     Start der Ein-/Ausgabebuffer
F5CC  21 FF 0F     LD     HL,0FFF       Länge -1
F5CF  09           ADD    HL,BC         addieren, gibt Endadresse
F5D0  CD E5 F5     CALL   F5E5          Datei-Endadr. im Buffer ?
F5D3  D1           POP    DE            Startadresse der Binärdatei
F5D4  DC E5 F5     CALL   C,F5E5        oder Datei-Start im Buffer ?
F5D7  D8           RET    C             nein ? dann o.k.
F5D8  EB           EX     DE,HL         Datei-Startadresse nach HL
F5D9  50           LD     D,B             Ein-/Ausgabebuffer-Start
F5DA  59           LD     E,C             nach DE
F5DB  CD F1 F5     CALL   F5F1          gleich HIMEM+1 ?
F5DE  C2 7F F7     JP     NZ,F77F       nein ? dann Buffer freigeben
F5E1  22 78 B0     LD     (B078),HL     Dateistart als Freigabe-HIMEM
F5E4  C9           RET
```

```
**********************************  Test, ob Adresse im Bereich liegt
                                    IN : BC: Startadresse d. Bereichs
                                         HL: Endadresse des Bereichs
                                         DE: zu testende Adresse
                                    OUT: CY=0, wenn im Bereich
F5E5  D5          PUSH   DE
F5E6  E5          PUSH   HL
F5E7  B7          OR     A               Start - Ende
F5E8  ED 42       SBC    HL,BC           End-Offset zu Bereichsstart
F5EA  EB          EX     DE,HL         End-Offset nach DE
F5EB  B7          OR     A               Offset der zu testenden Adr.
F5EC  ED 42       SBC    HL,BC           zum Bereichsstart
F5EE  EB          EX     DE,HL         nach DE, End-Offset nach HL
F5EF  18 06       JR     F5F7          Offsets vergleichen

**********************************  Adresse mit HIMEM+1 vergleichen
                                    IN : DE: zu vergleichende Adresse
                                    OUT: CY=1, wenn Adresse>HIMEM+1
                                         CY=0, Z=1, wenn Adr.=HIMEM+1
                                         CY=0, Z=0, wenn Adr.<HIMEM+1
F5F1  D5          PUSH   DE
F5F2  E5          PUSH   HL
F5F3  2A 5E AE    LD     HL,(AE5E)     HIMEM-Zeiger
F5F6  23          INC    HL            +1 gibt erste freie Adresse
F5F7  CD D8 FF    CALL   FFD8          mit DE vergleichen
F5FA  E1          POP    HL
F5FB  D1          POP    DE
F5FC  C9          RET

**********************************  (Prg.-/Var.-Zeiger korrigieren)
                                    IN : BC: Korrektur-Offset
F60C  2A 66 AE    LD     HL,(AE66)       Offset
F60F  09          ADD    HL,BC           zu Programmende
F610  22 66 AE    LD     (AE66),HL       addieren
F613  3A 6E AE    LD     A,(AE6E)      Variablenbereich
F616  B7          OR     A             geschützt ?
F617  C0          RET    NZ            dann Var.-Zeiger nicht korrig.

**********************************  Variablenbereich schützen
F62E  CD 08 F7    CALL   F708          Größe des freien Platzes
F631  44          LD     B,H             nach BC
F632  4D          LD     C,L
F633  2A 66 AE    LD     HL,(AE66)     Zeiger auf Programmende
F636  EB          EX     DE,HL         nach DE
F637  CD C4 F6    CALL   F6C4          dort maximalen Platz schaffen
F63A  3E FF       LD     A,FF          Flag für Variablen geschützt
F63C  18 0E       JR     F64C          setzen

**********************************  Variablenbereich ungeschützt
F63E  2A 66 AE    LD     HL,(AE66)     Zeiger auf Programmende
F641  EB          EX     DE,HL         nach DE
F642  2A 68 AE    LD     HL,(AE68)     Zeiger auf Variablenstart
F645  CD E4 FF    CALL   FFE4          Differenz nach BC
F648  CD F1 F6    CALL   F6F1          Bereich löschen
F64B  AF          XOR    A             Flag f. Variablen nicht gesch.
F64C  32 6E AE    LD     (AE6E),A      Flag setzen
F64F  C3 18 F6    JP     F618          Prg.-/Var.-Zeiger korrigieren
```

```
*********************************** Stringbereich-Platz reservieren
                                    IN : C: Läne des Strings
                                    OUT: HL: Adresse des Platzes
F696  06 00       LD     B,00          Länge hi =0
F698  2A 6C AE    LD     HL,(AE6C)     Ende der Arrays
F69B  EB          EX     DE,HL         nach DE
F69C  2A 71 B0    LD     HL,(B071)     Start der Strings
F69F  B7          OR     A
F6A0  ED 42       SBC    HL,BC         minus Stringlänge
F6A2  2B          DEC    HL              minus 2 für Längen-
F6A3  2B          DEC    HL              bzw. Descr.-Adr.-Eintrag
F6A4  CD D8 FF    CALL   FFD8          mit Ende d. Arrays vergleichen
F6A7  30 09       JR     NC,F6B2       genügend Platz ?
F6A9  CD 64 FC    CALL   FC64          sonst Garbage collection
F6AC  38 EA       JR     C,F698        neuen Platz geschaffen ?
F6AE  CD 48 CB    CALL   CB48          sonst Fehler melden
F6B1  0E                               Nr. f. "String space full"
F6B2  22 71 B0    LD     (B071),HL     neuen Start der Strings setzen
F6B5  23          INC    HL
F6B6  71          LD     (HL),C          Stringlänge
F6B7  23          INC    HL              eintragen
F6B8  70          LD     (HL),B
F6B9  23          INC    HL            Zeiger auf Platz für String
F6BA  C9          RET

*********************************** (Platz f. Programm/Var. schaffen)
                                    IN/OUT: DE: Einfügeadresse
                                            BC: benötigte Länge
                                    OUT: HL: neues Ende der Arrays
F6BB  3A 6E AE    LD     A,(AE6E)      Flag für Variablen geschützt
F6BE  B7          OR     A
F6BF  2A 66 AE    LD     HL,(AE66)     Zeiger auf Programmende
F6C2  20 03       JR     NZ,F6C7       Variablen geschützt ?
F6C4  2A 6C AE    LD     HL,(AE6C)     Zeiger auf Ende der Arrays
F6C7  C5          PUSH   BC
F6C8  D5          PUSH   DEF6C9  D5           PUSH   DE
F6CA  E5          PUSH   HL

*********************************** Bereich löschen
                                    IN : DE: Löschadresse
                                         BC: Länge
                                    OUT: BC: Offset
F6F1  C5          PUSH   BC
F6F2  D5          PUSH   DE
F6F3  EB          EX     DE,HL           Länge zu Startadresse des
F6F4  09          ADD    HL,BC           Löschbereichs addieren, gibt
F6F5  EB          EX     DE,HL           Lösch-Endadresse
F6F6  2A 6C AE    LD     HL,(AE6C)     Ende der Arrays
F6F9  CD E4 FF    CALL   FFE4          - Löschendadresse = Löschlänge
F6FC  EB          EX     DE,HL         End-Löschadresse als Quelle
F6FD  D1          POP    DE            Start-Löschadresse als Ziel
F6FE  CD EF FF    CALL   FFEF          Bereich verschieben
F701  D1          POP    DE            Lösch-Länge
F702  21 00 00    LD     HL,0000         Null minus Löschlänge
F705  C3 E4 FF    JP     FFE4            gibt Offset für Var.-Korr.
```

```
**********************************  höchste freie Adr. nach Prg. holen
                                    OUT: HL: höchste freie Adresse
F713  3A 6E AE     LD     A,(AE6E)     Flag für Variablen geschützt
F716  B7           OR     A
F717  2A 71 B0     LD     HL,(B071)    Start der Strings
F71A  C8           RET    Z            Variablen nicht geschützt ?
F71B  2A 68 AE     LD     HL,(AE68)    sonst Start der Variablen
F71E  2B           DEC    HL           -1 = höchste freie Adresse
F71F  C9           RET

**********************************  Eingabebuffer belegen
                                    OUT: DE: Zeiger auf Eingabebuffer
F720  11 01 00     LD     DE,0001      Offset und Flag f. Eingabebuf.
F723  18 08        JR     F72D

**********************************  Ausgabebuffer belegen
                                    OUT: DE: Zeiger auf Ausgabebuffer
F725  11 02 08     LD     DE,0802      Offset und Flag f. Eingabebuf.
F728  18 03        JR     F72D

**********************************  E/A-Buffer reservieren
                                    OUT: DE: Zeiger auf Ausgabebuffer
F72A  11 00 08     LD     DE,0800      Offset f. Ausgabebuffer
F72D  C5           PUSH   BC
F72E  E5           PUSH   HL
F72F  3A 75 B0     LD     A,(B075)     Buffer-Flags
F732  B7           OR     A
F733  20 18        JR     NZ,F74D      Buffer schon reserviert ?
F735  D5           PUSH   DE           Offset und Flag retten
F736  2A 5E AE     LD     HL,(AE5E)    HIMEM-Zeiger
F739  23           INC    HL           +1 gibt erste freie Adresse
F73A  22 78 B0     LD     (B078),HL    HIMEM+1 für Freigabe retten
F73D  11 00 F0     LD     DE,F000      - $1000 (Platz für 2 Buffer)
F740  19           ADD    HL,DE        addieren
F741  D2 75 F8     JP     NC,F875      Unterlauf ? dann kein Platz
F744  CD 08 F8     CALL   F808         HIMEM neu setzen
F747  22 76 B0     LD     (B076),HL    Adresse der E/A-Buffer setzen
F74A  D1           POP    DE           Offset/Flag zurück
F74B  3E 04        LD     A,04         Flag für Buffer reserviert
F74D  B3           OR     E            Belegungsflags setzen
F74E  2A 76 B0     LD     HL,(B076)    Adresse der Buffer
F751  1E 00        LD     E,00         Offset hi=0
F753  19           ADD    HL,DE        Offset addieren
F754  EB           EX     DE,HL        Adresse des Buffers nach DE
F755  E1           POP    HL
F756  C1           POP    BC
F757  18 27        JR     F780         Buffer-Flags neu setzen

**********************************  Teilstring ausgeben
                                    IN : String im FAC
                                         C: gewünschte Länge
                                    OUT: nicht ausgegebene Länge
F8DC  CD F5 FB     CALL   FBF5         String aus Str.-Ber/-Stack lö.
F8DF  C8           RET    Z            Länge =0 ?
F8E0  79           LD     A,C          gewünschte Länge
F8E1  90           SUB    B            minus tatsächliche Länge
F8E2  30 05        JR     NC,F8E9      gewünschte Länge zu groß ?
F8E4  80           ADD    B            gewünschte Länge wiederherst.
```

```
F8E5  28 02      JR    Z,F8E9       Flag für Gesamtstring ausg. ?
F8E7  47         LD    B,A          sonst gewünschte Länge
F8E8  AF         XOR   A            kein Zeichen nicht ausgegeben
F8E9  4F         LD    C,A          nicht ausgegebene Länge
F8EA  18 E8      JR    F8D4         String ausgeben

*********************************  zwei Strings vom Stringstack
                                   IN : HL: Adr. des 1. Descriptors
                                            2. Descriptor im FAC
                                   OUT: HL: Adresse des 1. Strings
                                         B: Länge des 1. Strings
                                        DE: Adresse des 2. Strings
                                         C: Länge des 2. Strings
F959  CD F5 FB   CALL  FBF5         2. String vom Stringstack
F95C  48         LD    C,B          Länge nach C
F95D  D5         PUSH  DE           Adresse retten
F95E  CD 03 FC   CALL  FC03         2. String vom Stringstack
F961  EB         EX    DE,HL        Adresse nach HL
F962  D1         POP   DE           Adresse des 2. Strings nach DE
F963  C9         RET

*********************************  Basic-Funktion INKEY$
FA7E  CD 72 C4   CALL  C472           Zeichen von Tastatur lesen
FA81  30 F5      JR    NC,FA78        keine Taste gedrückt ?
FA83  FE FC      CP    FC             ESC ?
FA85  28 F1      JR    Z,FA78         dann Leerstring
FA87  FE EF      CP    EF             BRK-Code ?
FA89  28 ED      JR    Z,FA78         dann Leerstring
FA8B  18 EA      JR    FA77           1-Zeichen-String generieren

*********************************  FAC nach Byte/1. Stringzeichen
                                   OUT: A: Bytewert/Zeichen
FAA1  CD 66 FF   CALL  FF66           Typ des FAC nicht String ?
FAA4  20 33      JR    NZ,FAD9        dann FAC nach Byte wandeln
FAA6  CD F5 FB   CALL  FBF5           String vom Stringstack
FAA9  28 37      JR    Z,FAE2         Länge =0 ? dann Fehler
FAAB  1A         LD    A,(DE)         1. Zeichen des Strings laden
FAAC  C9         RET

*********************************  Test, ob Descript. im Stringstack
                                   OUT: CY=1, wenn Descr. im Stack
FC37  2A A0 B0   LD    HL,(B0A0)      aktuelle Stringdescriptoradr.
FC3A  3E 7D      LD    A,7D             größer als $B07D ?
FC3C  95         SUB   L                dann im Stringstack, CY=1
FC3D  3E B0      LD    A,B0             (Stringstack beginnt
FC3F  9C         SBC   H                bei $B07E)
FC40  C9         RET

*********************************  Garbage collection
                                   OUT: CY=1, wenn Platz geschaffen
FC64  E5         PUSH  HL
FC65  D5         PUSH  DE
FC66  C5         PUSH  BC
FC67  21 7E B0   LD    HL,B07E        Zeiger auf Stringstack
FC6A  18 0C      JR    FC78           Stringstack durchgehen
FC6C  7E         LD    A,(HL)           Stringlänge
FC6D  23         INC   HL
FC6E  4E         LD    C,(HL)           und Stringadresse aus
```

```
FC6F  23          INC   HL            Descriptor laden
FC70  46          LD    B,(HL)
FC71  EB          EX    DE,HL        Descriptor-Endzeiger nach DE
FC72  B7          OR    A            Länge <>0 ?
FC73  C4 E3 FC    CALL  NZ,FCE3      dann Descriptoradr. eintragen
FC76  EB          EX    DE,HL        Descriptor-Endzeiger nach HL
FC77  23          INC   HL           Zeiger auf nächsten Descriptor
FC78  ED 5B 7C B0 LD    DE,(B07C)    Ende des Stringstacks
FC7C  CD D8 FF    CALL  FFD8         erreicht ?
FC7F  20 EB       JR    NZ,FC6C      nein ? dann Stack weiterbearb.
FC81  11 E3 FC    LD    DE,FCE3      Routine f. Descr. eintragen
FC84  CD 97 DA    CALL  DA97         alle Stringvariablen durchg.
FC87  2A 73 B0    LD    HL,(B073)    Zeiger auf Ende der Strings
FC8A  E5          PUSH  HL           retten
FC8B  2A 71 B0    LD    HL,(B071)    Zeiger auf Start der Strings
FC8E  23          INC   HL           Zeiger auf 1. benutztes Byte
FC8F  5D          LD    E,L            nach DE
FC90  54          LD    D,H            als Zieladresse
FC91  18 14       JR    FCA7         Strings durchgehen
FC93  4E          LD    C,(HL)         Descriptoradresse
FC94  23          INC   HL             nach BC
FC95  46          LD    B,(HL)
FC96  04          INC   B              Descriptoradresse
FC97  05          DEC   B              nicht eingetragen ?
FC98  28 0B       JR    Z,FCA5       dann nächsten String
FC9A  2B          DEC   HL           Zeiger auf Stringeintrag
FC9B  0A          LD    A,(BC)       Länge des String aus Descr.
FC9C  4F          LD    C,A          nach C
FC9D  06 00       LD    B,00         Länge hi =0
FC9F  03          INC   BC             Länge +2 für zusätzlichen
FCA0  03          INC   BC             2-Byte-Eintrag
FCA1  ED B0       LDIR               String nach unten schieben
FCA3  18 02       JR    FCA7         nächster String
FCA5  23          INC   HL           Zeiger auf String
FCA6  09          ADD   HL,BC        Länge addieren
FCA7  C1          POP   BC
FCA8  C5          PUSH  BC           Ende der Strings
FCA9  CD DE FF    CALL  FFDE         mit laufender Adresse vergl.
FCAC  38 E5       JR    C,FC93       Ende noch nicht erreicht ?
FCAE  1B          DEC   DE           letztes benutztes Stringbyte
FCAF  2A 71 B0    LD    HL,(B071)    Start der Strings (-1)
FCB2  EB          EX    DE,HL        nach DE, Ende nach HL
FCB3  CD E4 FF    CALL  FFE4         Differenz (Länge) nach BC
FCB6  D1          POP   DE           altes Ende der Strings
FCB7  CD D8 FF    CALL  FFD8         mit neuem vergleichen
FCBA  F5          PUSH  AF           Flag für zusätzl. Platz retten
FCBB  D5          PUSH  DE           altes Ende der Strings retten
FCBC  CD F5 FF    CALL  FFF5         Strings wieder n. oben schieb.
FCBF  EB          EX    DE,HL        Startzieladresse
FCC0  22 71 B0    LD    (B071),HL    als neuen Start der Strings
FCC3  C1          POP   BC           Ende der Strings
FCC4  23          INC   HL           neues 1. Stringbyte
FCC5  18 12       JR    FCD9         Strings durchgehen
FCC7  5E          LD    E,(HL)         Descriptorzeiger
FCC8  23          INC   HL             nach DE
FCC9  56          LD    D,(HL)
FCCA  2B          DEC   HL
FCCB  1A          LD    A,(DE)         Länge
```

```
FCCC  77          LD    (HL),A       wieder vor String setzen
FCCD  23          INC   HL
FCCE  36 00       LD    (HL),00     Länge hi=0
FCD0  23          INC   HL          Stringadresse
FCD1  EB          EX    DE,HL       nach DE
FCD2  72          LD    (HL),D        wieder in Descriptor
FCD3  2B          DEC   HL            eintragen
FCD4  73          LD    (HL),E
FCD5  6F          LD    L,A         Länge
FCD6  26 00       LD    H,00        Länge hi=0
FCD8  19          ADD   HL,DE       Länge addieren
FCD9  CD DE FF    CALL  FFDE        Ende der Strings
FCDC  38 E9       JR    C,FCC7      noch nicht erreicht ?
FCDE  F1          POP   AF          Flag für zusätzlichen Platz
FCDF  C1          POP   BC
FCE0  D1          POP   DE
FCE1  E1          POP   HL
FCE2  C9          RET

**********************************  Descriptoradresse eintragen
                                    IN : BC: Stringadresse
                                           DE: Descriptor-Endadresse
FCE3  2A 6C AE    LD    HL,(AE6C)   Ende der Arrays
FCE6  CD DE FF    CALL  FFDE        String unterh. Stringbereich ?
FCE9  D0          RET   NC          dann fertig
FCEA  0B          DEC   BC          Zeiger auf Länge hi
FCEB  7A          LD    A,D           Descriptoradresse hi
FCEC  02          LD    (BC),A        eintragen
FCED  0B          DEC   BC          Zeiger auf Länge lo
FCEE  0A          LD    A,(BC)      Länge
FCEF  12          LD    (DE),A      in Descriptor speichern
FCF0  7B          LD    A,E           Descriptor-Adresse lo
FCF1  02          LD    (BC),A        eintragen
FCF2  C9          RET

**********************************  Block nach unten verschieben
                                    (nur, wenn Länge<>0)
                                    IN : HL: Zeiger auf Quellblock
                                           DE: Zeiger auf Zielblock
                                           A: Länge
                                    OUT: HL: Zeiger nach Quellblock
                                           DE: Zeiger nach Zielblock
                                           A: wie IN
                                           BC: immer 0
FFEC  4F          LD    C,A         Länge
FFED  06 00       LD    B,00        nach BC

**********************************  Block nach unten verschieben
                                    (nur, wenn Länge<>0)
                                    IN : HL: Zeiger auf Quellblock
                                           DE: Zeiger auf Zielblock
                                           BC: Länge
                                    OUT: HL: Zeiger nach Quellblock
                                           DE: Zeiger nach Zielblock
                                           BC: immer 0
FFEF  78          LD    A,B
FFF0  B1          OR    C             Länge =0 ?
FFF1  C8          RET   Z           dann zurück
```

```
FFF2  ED B0         LDIR                  sonst Block verschieben
FFF4  C9            RET

*********************************** Block nach oben verschieben
                                    (nur, wenn Länge<>0)
                                    IN : HL: Zeiger a. Quellblockende
                                         DE: Zeiger auf Zielblockende
                                         BC: Länge
                                    OUT: HL: Zeiger vor Quellblock
                                         DE: Zeiger vor Zielblock
                                         BC: immer 0
FFF5  78            LD      A,B
FFF6  B1            OR      C                Länge =0 ?
FFF7  C8            RET     Z             dann zurück
FFF8  ED B8         LDDR                  sonst Block verschieben
FFFA  C9            RET
```

6.3 Die Listings des CPC-6128-ROMs

Das ROM des CPC 6128 ist gegenüber dem des CPC 664 nur an einigen Stellen geändert. An circa 20 Stellen treten durch Einfügungen oder Weglassungen Verschiebungen auf. Die relevanten Änderungen werden im Folgenden gelistet.

6.3.1 Das CPC 6128 - Betriebssystem

Das 6128-Betriebssystem ist (wie schon vom 464 und 664 her bekannt) in Packs aufgeteilt, die bei folgenden Adressen liegen:

1. Kernel (KL) $0000
2. Machine Pack (MC) $0591
3. Jump Restore $08BD
4. Screen Pack (SCR) $0ABF
5. Text Screen Pack (TXT) $1074
6. Graphics Screen Pack (GRA) $15A8
7. Keyboard Manager (KM) $1B5C
8. Sound Manager (SOUND) $1FE9
9. Cassette Manager (CAS) $24BC
10. Editor (EDIT) $2C02
11. Floating Point Arithmetics (FLO) $2F7D (Zeichensatz $3800)

```
***********************************   KL RAM SELECT
                                      IN : A: neue RAM-Konfiguration
                                      OUT: A: alte RAM-Konfiguration
0397  F3          DI
0398  D9          EXX
0399  21 D5 B8    LD     HL,B8D5        Adresse für RAM-Konfiguration
039C  56          LD     D,(HL)         alte Konfiguration laden
039D  77          LD     (HL),A         neue Konfiguration setzen
039E  F6 C0       OR     C0             Register 3 des Gate Array
03A0  ED 79       OUT    (C),A          Konfig. an Gate Array überg.
03A2  7A          LD     A,D            alte Konfig. nach A
03A3  D9          EXX
03A4  FB          EI
03A5  C9          RET

***********************************   (MC START PROGRAM)
062D  01 C0 7F    LD     BC,7FC0        RAM-Konfiguration 0
0630  ED 49       OUT    (C),C          ans Gate Array übergeben
0632  01 7E FA    LD     BC,FA7E        Disk-Schnittstelle
0635  AF          XOR    A                zurück-
0636  ED 79       OUT    (C),A            setzen

***********************************   Einschaltmeldung
0688  20 31 32 38 4B 20 4D 69           128K Mi
0690  63 72 6F 63 6F 6D 70 75          crocompu
0698  74 65 72 20 20 28 76 33          ter  (v3
06A0  29 1F 02 04 43 6F 70 79          )...Copy
06A8  72 69 67 68 74 1F 02 04          right...
06B0  A4 31 39 38 35 20 41 6D          .1985 Am
06B8  73 74 72 61 64 20 43 6F          stard Co
06C0  6E 73 75 6D 65 72 20 45          nsumer E
06C8  6C 65 63 74 72 6F 6E 69          lectroni
06D0  63 73 20 70 6C 63 1F 0C          cs plc..
06D8  05 61 6E 64 20 4C 6F 63          .and Loc
06E0  6F 6D 6F 74 69 76 65 20          omotive
06E8  53 6F 66 74 77 61 72 65          Softwrae
06F0  20 4C 74 64 2E 1F 01 07          ltd....
06F8  00                               .

***********************************   (Jump-Restore-Vekt., Haupttab.)
0A70  97 03                             KL RAM SELECT (über BD5B)

***********************************   (CAS IN OPEN)24E5  DD 21 1A B1  LD     IX,B11A
24E9  CD 02 25    CALL   2502           File öffnen
24EC  E5          PUSH   HL
24ED  DC AC 26    CALL   C,26AC         kein Fehler? d. 1. Block lesen
24F0  E1          POP    HL
24F1  D0          RET    NC             File- oder logischer Fehler
24F2  ED 5B 34 B1 LD     DE,(B134)        sonst Parameter
24F6  ED 4B 37 B1 LD     BC,(B137)        laden
24FA  3A 31 B1    LD     A,(B131)
24FD  C9          RET
***********************************   CRTL-TAB im Editor
2D81  3A 15 B1    LD     A,(B115)       Insert-Flag
2D84  2F          CPL                   invertieren
2D85  32 15 B1    LD     (B115),A       und neu setzen
2D88  B7          OR     A              CY=0, da kein Abbruch
2D89  C9          RET
```

6.3.2 Das Basic des CPC 6128

```
**********************************   Variablenbereich wieder ungeschü.
F63C  AF          XOR   A              Flag für Var. nicht geschützt
F63D  32 6E AE    LD    (AE6E),A       setzen
F640  2A 66 AE    LD    HL,(AE66)      Zeiger auf Programmende
F643  EB          EX    DE,HL          nach DE
F644  2A 68 AE    LD    HL,(AE68)      Zeiger auf Variablenstart
F647  CD E4 FF    CALL  FFE4           Differenz nach BC
F64A  CD E5 F6    CALL  F6E5           Bereich löschen
F64D  18 C4       JR    F613           Prg.-/Var.-Zeiger korrigieren

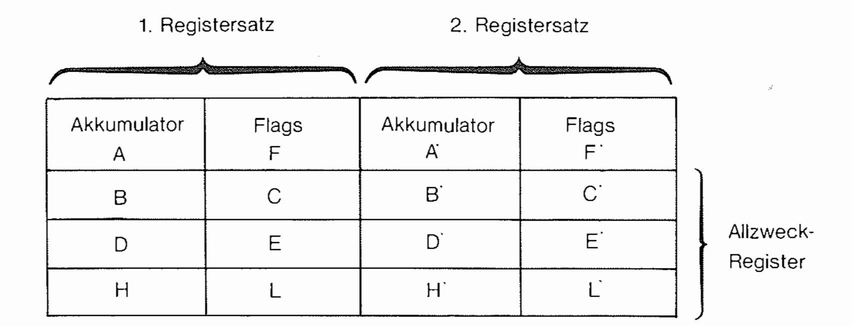
**********************************   (Bereich löschen)
F6E9  EB          EX    DE,HL
F6EA  CD 14 F7    CALL  F714           Ende der Arrays holen
F6ED  CD E4 FF    CALL  FFE4
F6F0  EB          EX    DE,HL

**********************************   Zeiger auf freien Basic-Bereich holen
                                     OUT: HL: Zeiger
F714  3A 6E AE    LD    A,(AE6E)       Flag für Variablen geschützt
F717  B7          OR    A
F718  2A 6C AE    LD    HL,(AE6C)      Zeiger auf Ende des Arrays
F71B  C8          RET   Z              Variablen nicht geschützt ?
F71C  2A 66 AE    LD    HL,(AE66)      sonst Zeiger auf Programmende
F71F  C9          RET
```

A Anhang

A1 Z80A CPU

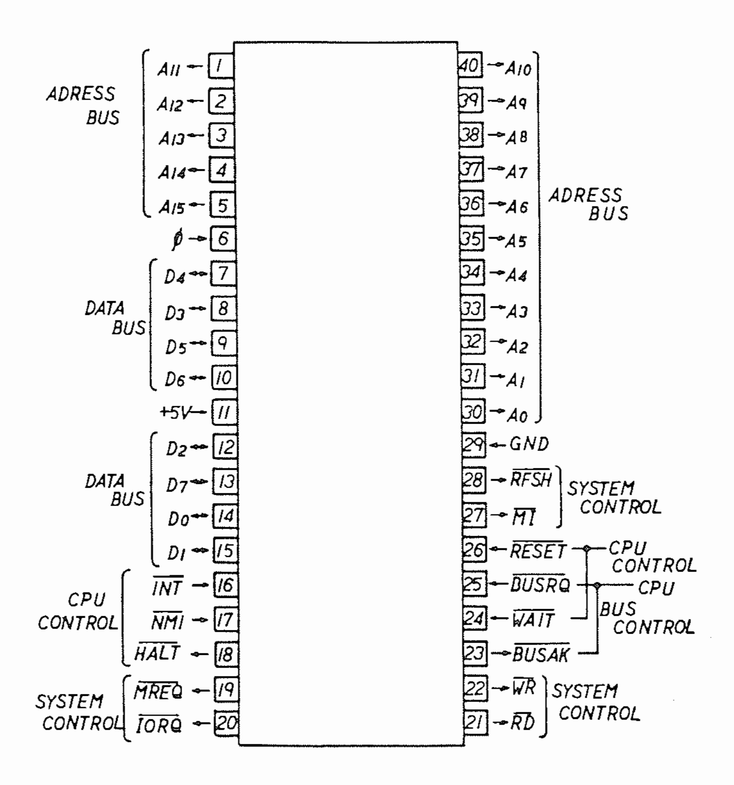
A1.1 Register



Interrupt Vector I	Memory Refresh R
Index Register IX	
Index Register IY	
Stack Pointer SP	
Program Counter PC	

Besondere Register

A1.2 Pin Out*



* Quelle: Kundendienst Handbuch Service Manual, Schneider Computer Division

A1.3 Befehlstabellen

Die Z80A-Befehle werden in folgende Gruppen aufgeteilt:

8-Bit-Ladebefehle	Akkumulator- und Flag-Operationen
16-Bit-Ladebefehle	Verschiedenes
Vertausch-Befehle	Rotier- und Schiebebefehle
Block-Verschiebe-Befehle	Bit-Operationen
Block-Suchbefehle	Ein-/Ausgabe
8-Bit-Arithmetik und -Logik	Sprünge
16-Bit-Arithmetik	Unterprogramm-Behandlung

In der Tabelle wird folgende Terminologie benutzt:

S = Sign-Flag (Vorzeichen bzw. höchstes Bit)
Z = Zero-Flag (Z=1, wenn Ergebnis gleich Null)
H = Halfcarry-Flag (Übertrag vom 3. zum 4. Bit)
P/V = Parity-/Overflow-Flag (Parität/Überlauf)
N = Negative-Flag (N=1, wenn Subtraktion vorausging)
CY = Carry-Flag (Übertrag zum nächsten Byte)
b = Nummer eines Bits in einem 8-Bit-Wert
cc = Bedingung abhängig vom Zustand der Flags
- NZ = ungleich Null (Z=0)
- Z = gleich Null (Z=1)
- NC = kein Übertrag (CY=0)
- C = Übertrag (CY=1)
- PO = ungerade Parität oder kein Überlauf (P/V=0)
- PE = gerade Parität oder Überlauf (P/V=1)
- P = positiv (höchstes Bit =0) (S=0)
- N = negativ (höchstes Bit =1) (S=1)

d = 8-Bit-Zielregister oder -Speicherstelle
dd = 16-Bit-Zielregister oder -Speicherstelle
e = 8-Bit-Wert im Zweierkomplement
L = Spezielle Sprungadresse (hex. 00,08,10,18,20,28,30,38)
n = 8-Bit-Wert, direkt im Programm folgend
nn = 16-Bit-Wert, direkt im Programm folgend
r = 8-Bit-Allzweck-Register (A,B,C,D,E,H,L)
s = 8-Bit-Quellregister oder -Speicherstelle
s_b = ein Bit eines Registers oder einer Speicherstelle
ss = 16-Bit-Quellregister oder -Speicherstelle
Index "L" = die 8 niederwertigen Bits eines 16-Bit-Werts (Low-Byte)
Index "H" = die 8 höherwertigen Bits eines 16-Bit-Werts (High-Byte)
( ) = der Wert in den Klammern ist eine Speicher- oder Ein-/Ausgabe-adresse, deren Inhalt gelesen oder beschrieben wird

8-Bit-Register sind A,B,C,D,E,H,L,I und R
16-Bit-Registerpaare sind AF, BC, DE und HL
16-Bit-Register sind SP, PC, IX, und IY

Folgende Adressierungsarten werden verwendet beziehungsweise miteinander kombiniert:

Unmittelbar (Immediate)	Indiziert (Indexed)
Immediate extended (16-Bit-Werte)	Register
modifizierte Zero-Page-Adressierung	Implizit
Relativ	Register indirekt
Extended	Bitweise

	Mnemonic	symbolische Beschreibung	Bemerkungen
8-Bit-Ladebefehle	LD r, s	r ← s	s = r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d = (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d = (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s = (BC), (DE), (nn), I, R
	LD d, A	d ← A	d = (BC), (DE), (nn), I, R
16-Bit-Ladebefehle	LD dd, nn	dd ← nn	dd = BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd = BC, DE, HL, SP, IX, IY
	LD (nn) ss	(nn) ← ss	ss = BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	(SP-I) ← ss_H;(SP-2) ← ss_L	ss = BC, DE, HL, AF, IX, IY
	POP dd	dd_L ← (SP);dd_H ← (SP+1)	dd = BC, DE, HL, AF, IX, IY
Austauschbefehle	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	(BC, DE, HL) ↔ (BC', DE', HL')	
	EX (SP), ss	(SP) ↔ ss_L, (SP+1) ↔ ss_H	ss = HL, IX, IY

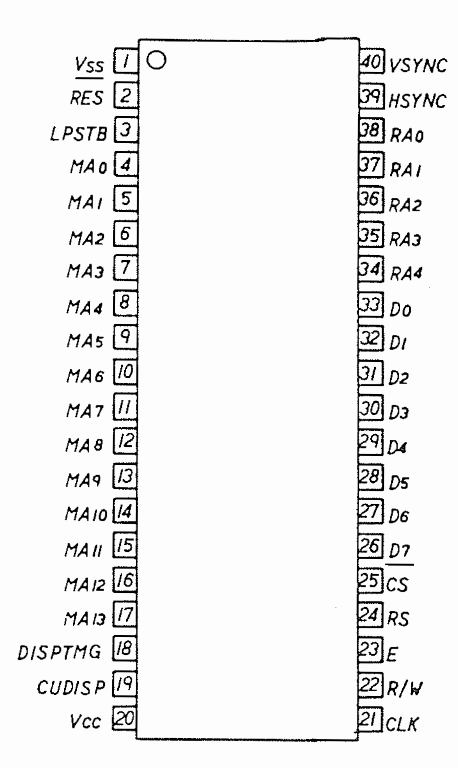
	Mnemonic	symbolische Beschreibung	Bemerkungen
Blockverschiebungen	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 wiederholen, bis BC=0	
	LDD	(DE) ← (HL), DE ← DE-1 HL ← HL-1,BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1 wiederholen, bis BC=0	
Blocksuch-Befehle	CPI	A-(HL), HL ← HL+1 BC ← BC-1	A-(HL) setzt nur die Flags, A wird nicht verändert
	CPIR	A-(HL), HL ← HL+1 BC ← BC-1, wiederholen, bis BC=0 oder A=(HL)	
	CPD	A-(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1 BC ← BC-1, wiederholen, bis BC=0 oder A=(HL)	
8-Bit-Operationen	ADD s	A ← A+s	s = r, n, (HL) (IX+e), (IY+e)
	ADC s	A ← A+s+CY	
	SUB s	A ← A-s	
	SBC s	A ← A-s-CY	
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A⊕s	

	Mnemonic	symbolische Beschreibung	Bemerkungen
8-Bit-Operationen	CP s	A-s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d+1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d-1	
16-Bit-Arithmetik	ADD HL,ss	HL ← HL+ss	ss = BC, DE HL, SP
	ADC HL,ss	HL ← HL+ss+CY	
	SBC HL,ss	HL ← HL-ss-CY	
	ADD IX,ss	IX ← IX+ss	ss = BC, DE IX, SP
	ADD IY,ss	IY ← IY+ss	ss = BC, DE IY, SP
	INC dd	dd ← dd+1	dd = BC, DE HL, SP, IX, IY
	DEC dd	dd ← dd-1	dd = BC, DE HL, SP, IX, IY
Akkumulator/Flags	DAA	Korrigiert A nach einer Addition oder Subtraktion mit gepackten BCD-Zahlen	
	CPL	A ← $\overline{A}$	
	NEG	A ← 00-A	
	CCF	CY ← $\overline{CY}$	
	SCF	CY ← 1	
Verschiedenes	NOP	Keine Operation	
	HALT	Auf Interrupt warten	
	DI	Interrupts sperren	
	EI	Interrupts erlauben	Wirkung erst nach dem folgenden Befehl
	IM 0	Interrupt-Modus 0	8080A-Modus
	IM 1	Interrupt-Modus 1	CALL nach hex. 0038
	IM 2	Interrupt-Modus 2	Indirekter Aufruf
Rotier- und Schiebebefehle	RLC s		s = r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemonic	symbolische Beschreibung	Bemerkungen
Bitbefehle	BIT b, s	Z ← $\overline{s_b}$	
	SET b, s	s_b ← 1	s = r, (HL) (IX+e), (IY+e)
	RES b, s	s_b ← 0	
Ein-/Ausgabe	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Flags setzen
	INI	(HL) ← (C), HL ← HL+1 B ← B-1	
	INIR	(HL) ← (C), HL ← HL+1 B ← B-1 wiederhole, bis B=0	
	IND	(HL) ← (C), HL ← HL-1 B ← B-1	
	INDR	(HL) ← (C), HL ← HL-1 B ← B-1 wiederhole, bis B=0	
	OUT(n), A	(n) ← A	statt (C) eigentlich (BC), da BC auf den Adreßbus gelegt wird
	OUT(C),r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL+1 B ← B-1	
	OTIR	(C) ← (HL), HL ← HL+1 B ← B-1 wiederhole, bis B=0	
	OUTD	(C) ← (HL), HL ← HL-1 B ← B-1	
	OTDR	(C) ← (HL), HL ← HL-1 B ← B-1 wiederhole, bis B=0	
Sprünge	JP nn	PC ← nn	
	JP cc, nn	Wenn Bedingung cc erfüllt ist, PC ← nn, sonst fortfahren	cc: NZ PO, Z PE, NC P, C M
	JR e	PC ← PC+e	
	JR kk, e	Wenn Bedingung kk erfüllt ist, PC ← PC+e, sonst fortfahren	kk: NZ NC, Z C
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B-1, wenn B < > 0, PC ← PC+e, sonst fortfahren	
Unterprogramme	CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L, PC ← nn	
	CALL cc, nn	Wenn Bedingung cc erfüllt ist, CALL nn, sonst fortfahren	cc: NZ PO, Z PE, NC P, C M
	RST L	(SP-1) ← PC_H (SP-2) ← PC_L, PC_H ← 0 PC_L ← L	
	RET	PC_L ← (SP) PC_H ← (SP+1)	
	RET cc	Wenn Bedingung cc erfüllt ist, RET, sonst fortfahren	cc: NZ PO, Z PE, NC P, C M
	RETI	Rückkehr vom Interrupt, sonst wie RET	
	RETN	Rückkehr vom nicht-maskierbaren Interrupt	

A2 6845 CRTC

A2.1 Pin Out*



* Quelle: Kundendienst Handbuch Service Manual, Schneider Computer Division

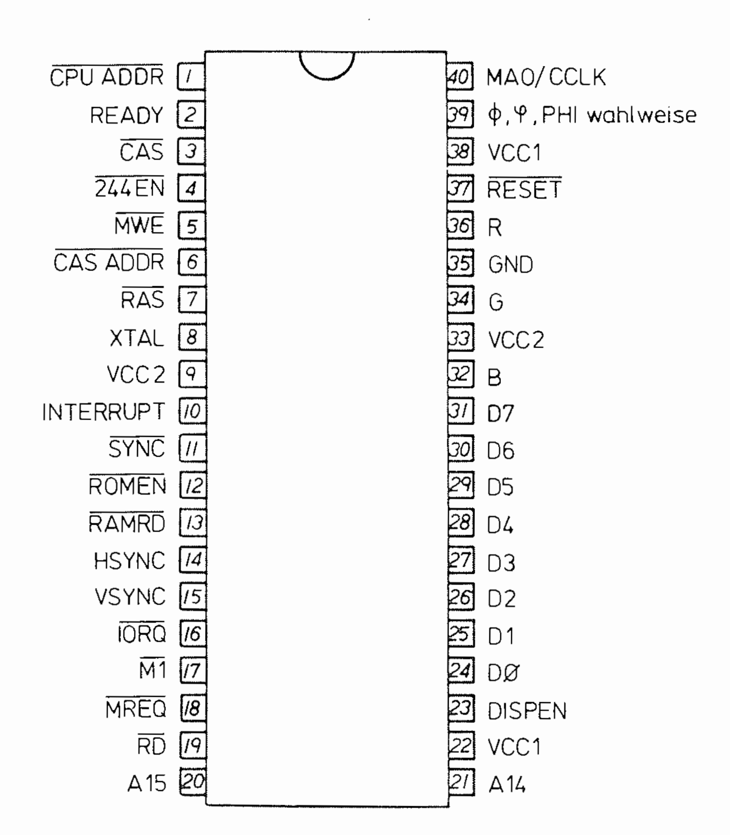
A2.2 Registerbelegung*

Register #	Register File	Program Unit	Read	Write	Number of Bits							
					7	6	5	4	3	2	1	0
X	–	–	–	–								
AR	Address Register	–	No	Yes								
R0	Horizontal Total	Char.	No	Yes								
R1	Horizontal Displayed	Char.	No	Yes								
R2	H. Sync Position	Char.	No	Yes								
R3	Sync Width	–	No	Yes								
R4	Vertical Total	Char. Row	No	Yes								
R5	V. Total Adjust	Scan Line	No	Yes								
R6	Vertical Displayed	Char. Row	No	Yes								
R7	V. Sync Position	Char. Row	No	Yes								
R8	Interlace Mode and Skew	Note 1	No	Yes								
R9	Max Scan Line Address	Scan Line	No	Yes								
R10	Cursor Start	Scan Line	No	Yes								
R11	Cursor End	Scan Line	No	Yes								
R12	Start Address (H)	–	No	Yes	0	0						
R13	Start Address (L)	–	No	Yes								
R14	Cursor (H)	–	Yes	Yes	0	0						
R15	Cursor (L)	–	Yes	Yes								
R16	Light Pen (H)	–	Yes	No	0	0						
R17	Light Pen (L)	–	Yes	No								

* Quelle: Motorola

A3 20 RA 43 Gate Array

A3.1 Pin Out

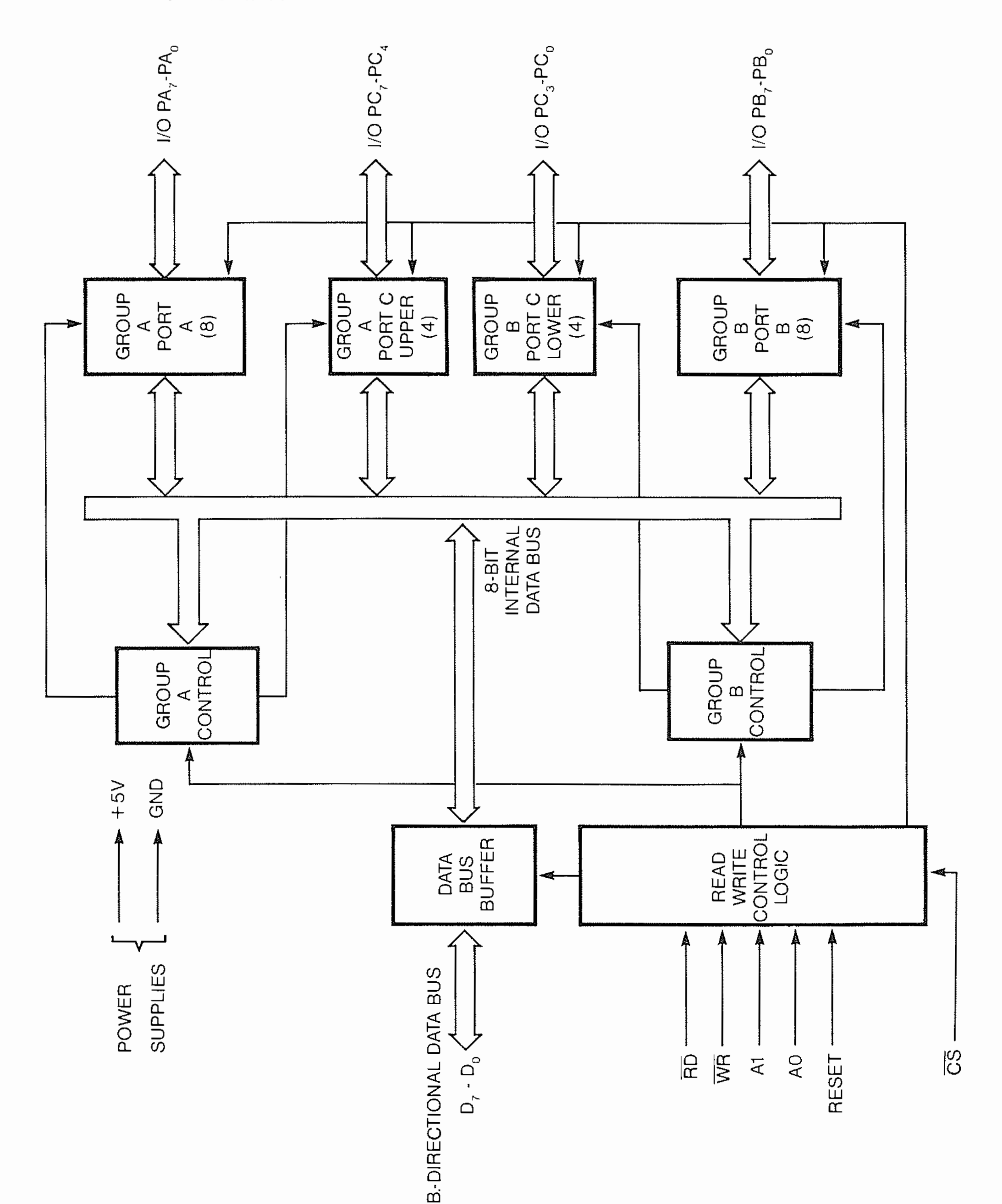


A3.2 Registerbelegung

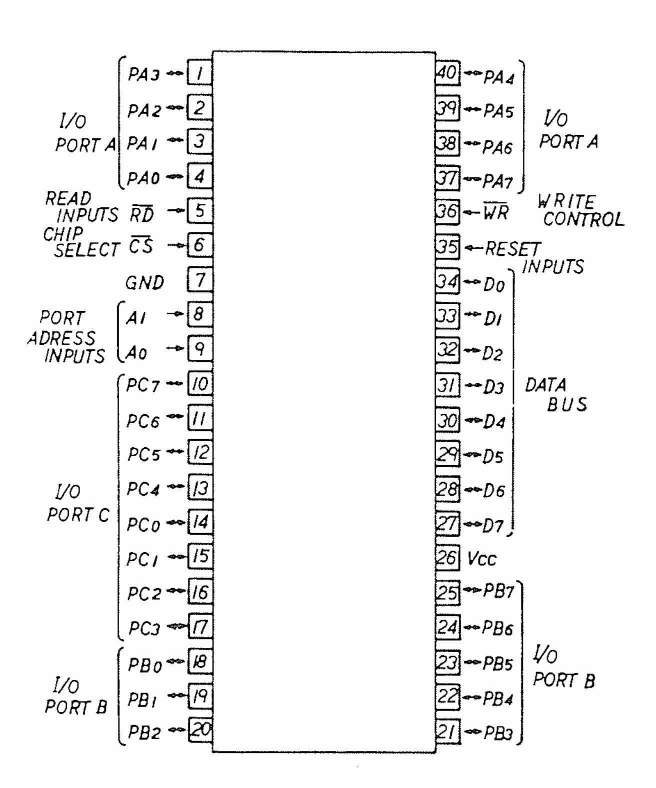
REG # (b7=0, b6=0): Farb-Adress-Register
REG # (b7=0, b6=1): Farbwert-Datenregister
REG # (b7=1, b6=0): Kontroll-Register
b4: 1: Interrupt-Zähler löschen
b3: 0: oberes ROM einschalten
b2: 0: unteres ROM einschalten
b1: 0: Mode-Auswahl
b0: 0: Mode-Auswahl
REG # (b7=1, b6=1): Funktion unbekannt

A4 8255 PIO

A4.1 Blockschaltbild



A4.2 Pin Out*

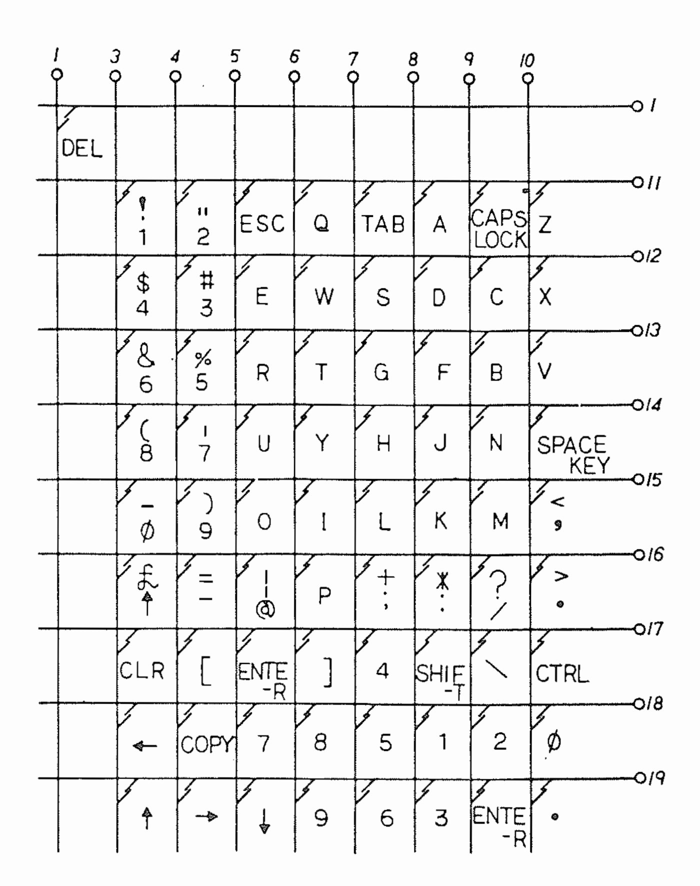


A4.3 Registerbelegung

A1	A0	I/O-Adresse	ausgewählte Einheit
0	0	$F4xx	Port A
0	1	$F5xx	Port B
1	0	$F6xx	Port C
1	1	$F7xx	Kontroll-Register

* Quelle: Kundendienst Handbuch Service Manual, Schneider Computer Division

A4.4 Tastaturmatrix*

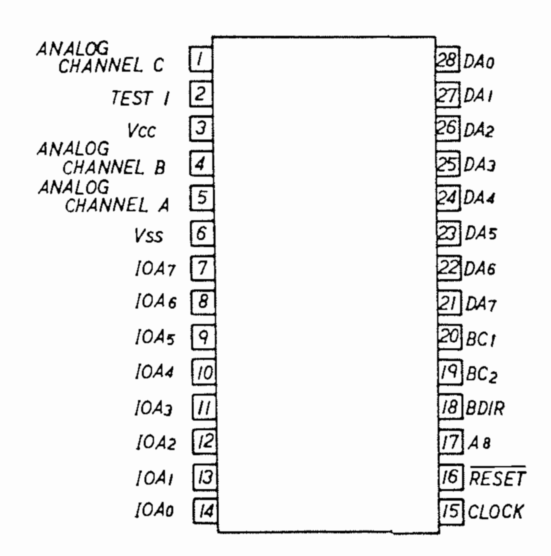


---

* Quelle: Kundendienst Handbuch Service Manual, Schneider Computer Division

A5 AY 3-8912 PSG

A5.1 Pin Out*

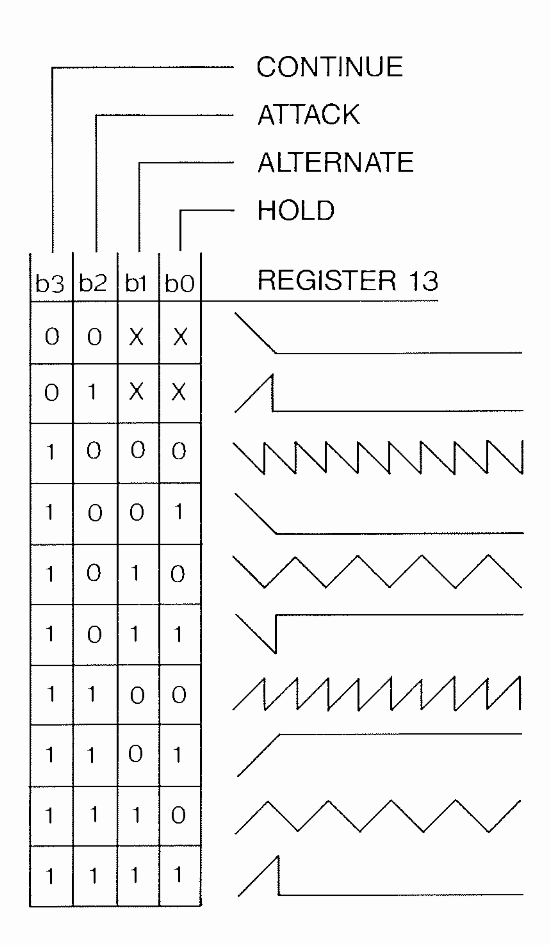


A5.2 Registerbelegung

REG #0/1:	Periodendauer Kanal A (12 Bit)
REG #2/3:	Periodendauer Kanal B (12 Bit)
REG #4/5:	Periodendauer Kanal C (12 Bit)
REG #6:	durchschnittliche Periodendauer Rauschen (5 Bit)
REG #7:	Kontroll-Register
REG #8:	Lautstärke Kanal A (4 Bit)
REG #9:	Lautstärke Kanal B (4 Bit)
REG #10:	Lautstärke Kanal C (4 Bit)
REG #11/12:	Periodendauer der Hüllkurve (16 Bit)
REG #13:	Hüllkurvenform (4 Bit)
REG #14:	peripheres Datenregister Port A
REG #15:	peripheres Datenregister Port B

* Quelle: Kundendienst Handbuch Service Manual, Schneider Computer Division

A5.3 Hüllkurventabelle



b3	b2	b1	b0
0	0	X	X
0	1	X	X
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Abkürzungsverzeichnis

In der Computerfachsprache herrscht allgemein ein hohes Aufkommen an Abkürzungen, und wir mußten darüber hinaus im Rahmen unserer Analyse des CPC auch noch einige Kürzel einführen, da die entsprechenden Termini einfach zu lang wurden. Deshalb halten wir es für sinnvoll, an dieser Stelle einmal die wichtigsten und gebräuchlichsten Abkürzungen zusammenzufassen.

Adr	Adresse
APQ	Asynchronous Pending Queue
Arg	Argument
ASCII	American Standard Code for Information Interchange, Standard für die Übertragung von Daten (USASCII)
async	asynchronous, asynchron
Basic	Beginners All Purpose Symbolic Instruction Code, eine Programmiersprache
Bit	Binary digit, Ziffer des binären Zahlensystems, 0 oder 1
BRK	BReaK-Zeichen, CPC-spezielles Zeichen, $EF
CAS	CASsette Manager Pack
Char	Character, Zeichen
CP/M	Control Program for Microcomputers, verbreiteter Standard für Betriebssysteme
CPC	Color Personal Computer: das Gerät.
CPU	Central Processing Unit, im Falle des CPC der Z80-Chip (vielfach auch für Prozessor inkl. ROM und RAM verwandt)
CR	Carriage Return, Wagenrücklauf, ASCII-Zeichen $0D
CRT	Cathode Ray Tube, Kathodenstrahlröhre
CRTC	Cathode Ray Tube Controller, der Video-Chip im CPC
CTRL	ConTRoL, Kontrolltaste
CY	CarrY, Übertrags-Flag, eines der Z80-Flags
Descr	Descriptor (eines Strings)
DMA	Direct Memory Access, direkter Zugriff externer Einheiten auf das zentrale RAM der CPU
DOS	Disk Operating System, Betriebssystem auf Diskettenbasis
E/A,EA	Ein-/Ausgabe
ENT	ENvelope Tone, Ton-Hüllkurve
ENV	ENvelope Volume, Lautstärke-Hüllkurve
EOF	End Of File, Ende einer Datei
ESC	ESCape, ASCII-Zeichen, $1B (im CPC $FC)
Exp	1. Exp String: Expansion String 2. Exponent (allgemein) 3. 8-Bit Exponent einer FLO-Fahl
FAC	Floating point ACcumulator, Speicherbereich für FLO-Zahl

FCFS	First Come - First Served, wie FIFO
FFC	Frame Fly Chain
FIFO	First In - First Out, Queue-Struktur
FLO	FLOating point, Fließkomma-...
FLR	Fixed Length Record, Record mit fester Länge
FTC	Fast Ticker Chain
GA	Gate Array, einer der Bausteine im CPC
GND	GrouND, Bezugsspannung innerhalb eines elektrischen Systems
GRA	GRAphics screen pack
HC	Hierarchie Code
Hex	Hexadezimal (korrekt eigentlich Sedezimal)
I/O,IO	Input/Output, Eingabe/Ausgabe
INT	INTeger, ganzzahlig
IPQ	Interrupt Pending Queue (entspricht APQ)
K	Faktor 1024 (nicht etwa k, Faktor 1000)
KA	Koppeladresse
KAFFC	Koppeladresse für die Frame Fly Chain
KAFTC	Koppeladresse für die Fast Ticker Chain
KAPQ	Koppeladresse für die Pending Queue
KATC	Koppeladresse für die Ticker Chain
KByte	1024 Byte
KL	KerneL
KM	Keyboard Manager Pack
LF	Line Feed, ASCII-Zeichen $0A
LIFO	Last In - First Out, Stack-Struktur
LL	Linked List, verkettete Liste
LSB	Least Significant Byte/Bit, niederwertigstes Byte/Bit
MC	MaChine Pack
MSB	Most Significant Byte/Bit, höchstwertiges Byte/Bit
NIL	Not In List, Markierung für Listenende
Param	Parameter
PC	Program Counter, eines der Z80-Register Basic PC: Programmzeiger innerhalb eines Basic-Programms
PIO	Programmable Input/Output Chip, ein Chip im CPC
PQ	Pending Queue
PSG	Programmable Sound Generator, ein Chip im CPC
PTR	PoinTeR, Zeiger
RAM	Random Access Memory, Schreib-/Lesespeicher
Reg,REG	Register
ROM	Read Only Memory, Nur-Lesespeicher
RSX	Resident System eXtension, residente Systemerweiterung
S	Sign-Flag, eines der Z80-Flags

SCR	SCReen Pack
SOUND	SOUND Manager Pack
SP	Stack Pointer: 1. eines der Z80-Register 2. allgemeiner Zeiger einer LIFO-Struktur
SPQ	Synchronous Pending Queue
SWI	SoftWare Interrupt, Prozessorbefehl des MC6809
sync	synchronous, synchron
TC	Ticker Chain
TXT	TeXT Pack
UCSD	University of California, San Diego
VDU	Video Display Unit, siehe CRT
VL	verkettete Liste
VLR	Variable Length Record
Z	Zero-Flag, eines der Z80-Flags

Stichwortverzeichnis